Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Boštjan Novak

Hišni avtomatizacijski sistem s podporo glasovnemu upravljanju

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Robert Rozman

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V današnjem času so sistemi za avtomatizacijo bivalnih okolij vse bolj popularni. Kljub temu je za njihovo upravljanje še vedno potrebna višja raven poznavanja tovrstnih sistemov, prav tako so običajno težje povezljivi in večinoma ne preveč splošni. Zato oblikujte splošnejši koncept avtomatiziranega bivalnega okolja, določite njegove osnovne gradnike, njihovo funkcionalnost ter podrobneje opredelite njihovo medsebojno interakcijo. Opisano zasnovo preizkusite v praksi; realizirajte sistem z osnovnimi konceptualnimi gradniki na konkretnih sistemih. Implementirajte tudi možnost preprostega upravljanja sistema s pomočjo govorne komunikacije med uporabnikom in sistemom.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Boštjan Novak sem avtor diplomskega dela z naslovom:

Hišni avtomatizacijski sistem s podporo glasovnemu upravljanju

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Roberta Rozmana,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 11. marca 2016 Podpis avtorja:

Zahvaljujem se mentorju, viš. pred. dr. Robertu Rozmanu, za svetovanje in pomoč pri izdelavi diplomskega dela. Zahvala velja tudi staršem za finančno pomoč ter vsem, s katerimi smo preživeli nepozabno študentsko obdobje.

Kazalo

Slike

Tabele

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hypertext Markup Language	Spletni označevalni jezik
HTML5	Hypertext Markup Language 5	Spletni označevalni jezik
CSS	Cascading Style Sheets	Kaskadne stilske podloge.
HTTP	HyperText Transfer Protocol	Protokol za prenos hiperteksta
HTTPS	HyperText Transfer Protocol Secure	Varni protokol za prenos hiperteksta
SQL	Structured Query Language	Strukturirani povpraševalni jezik
JSON	JavaScript Object Notation	JavaScript objektna notacija
XML	Extensible Markup Language	Razširljiv označevalni jezik
AJAX	Asynchronous JavaScript and XML	Asinhroni JavaScript in XML
REST	Representational State Transfer	Arhitekturni stil izmenjave podatkov
MCV	Model-View-Controller	Model-pogled-kontrola
GPIO	General-Purpose Input/Output	Vhodno-izhodni kontakti
PBKDF2	Password-Based Key Derivation Function 2	Kriptografski algoritem
USB	Universal Serial Bus	Univerzalno serijsko vodilo

Povzetek

V okviru diplomske naloge smo izdelali hierarhični sistem za avtomatizacijo, ki je sestavljen iz osrednjega strežnika ter posameznih povezanih odjemalcev. Delovanje obeh je vzajemno, saj se med njima vseskozi izmenjujejo zahteve, ki opisujejo stanja nanj priključenih zunanjih naprav. Z njimi lahko upravljamo preko uporabniškega vmesnika ali z glasovnim upravljanjem na strani odjemalca. Uporaba slednjega nam omogoča upravljanje naprav s pomočjo govora, kjer se nam kot povratni odziv predvaja odgovor v obliki umetno tvorjenega govora.

Naš končni izdelek je sestavljen iz spletne in odjemalne aplikacije, z uporabo katerih lahko v eno skupno mrežo povežemo različne tipe zunanjih naprav ali senzorjev. Za primarni preizkus odjemalne aplikacije smo uporabili Raspberry Pi računalnik, ki nam vse to omogoča s svojimi vgrajenimi GPIO nožicami. Z napravami lahko upravljamo ročno preko vmesnika ali spletni aplikaciji določimo časovni termin, v katerem naj ta samodejno aktivira izbrane naprave. Pri razvoju glasovnega upravljanja smo za razpoznavo ter sintezo govora uporabili storitvi Google Speech Recognition ter Google Synthesis. Z razpoznanim govorom v tekstovni obliki lahko nato upravljamo s priključenimi napravami ali ga pošljemo storitvi Wolfram Alpha, ki poskuša zanj poiskati primeren povratni odziv. Ta se po opravljenem postopku sinteze predvaja na odjemalcu preko priključenih zvočnikov.

Ključne besede: Hišni avtomatizirani sistem, glasovno upravljanje, razpoznava govora, sinteza govora.

Abstract

In the thesis, we have developed a hierarchical system for automation, which consists of a central server and connected clients. The operation of the two is mutual because they constantly exchange requests with latest states of connected devices. They can be managed through the user interface or by voice control on the client side. With the recognized speech, we can control specific connected devices and, as a result, get a response in the form of synthesized speech.

Our final product is composed of web and client applications that can connect different types of external devices or sensors in the unified network. For initial test of the client application, we decided to use the Raspberry Pi computer, which enables connection of external devices to its built-in GPIO pins. Those devices can be managed manually via the user interface or by web application – user can set the time when the application should automatically activate the selected devices. In the development of the voice control interface, internet services Google Speech Recognition and Synthesis were used. Using recognized speech content, we can manage connected devices or send text to service Wolfram Alpha; it tries to find an appropriate answer, that can be played as a synthesized speech on client using connected speakers.

Keywords: home automation system, voice control, speech recognition, speech synthesis.

Poglavje 1

Uvod

V današnjem času se pogosto pojavi potreba po učinkovitem sistemu, ki bi v okviru pametne hiše lahko v eno skupno mrežo povezoval različne tipe naprav in senzorjev. Uporabniki sistema bi lahko tako z priključenimi napravami upravljali kar preko spletnega vmesnika, ki bi na enem mestu združeval možnosti upravljanja ter hkrati v pregledni obliki prikazoval sveže informacije o delovanju celotnega sistema. Zanj bi bilo zaželjeno, da bi omogočal samodejni vklop naprav v časovnem intervalu, ki bi ga uporabniki določili sami preko grafičnega vmesnika. Tako bi uporabniki lahko sami določili časovni termin, v katerem bi se izbrane naprave aktivirale tudi brez njihove prisotnosti. Glede na dejstvo, da smo ljudje dovzetnejši za interaktivne načine upravljanja, smo se ob načrtovanju projekta odločili, da na strani odjemalnih naprav vključimo podporo glasovnemu upravljanju naprav. Tako bi lahko uporabniki kar s pomočjo govora vklapljali ali izklapljali posamezne naprave. Posledično bi to doprineslo k večjemu zanimanju za izdelek, kar bi ugodno vplivalo na tržni vidik sistema. Ker je celotna ideja ter izvedba projekta naše avtorsko delo, smo se posledično manj sklicevali na zunanje reference.

V okviru diplomskega dela smo izdelali spletno aplikacijo, ki predstavlja jedro sistema, ter odjemalno aplikacijo, ki skrbi za upravljanje priključenih naprav na strani odjemalcev. Z vidika celotnega sistema je njuno delovanje

vzajemno, saj med njima skozi poteka izmenjava zahtev in podatkov. Na strani spletne aplikacije smo zasnovali tudi hiearhično strukturo aparaturnih gradnikov, ki med drugim omogoča medsebojno povezavo odjemalcev ter nanje priključenih naprav. Slednjo smo zasnovali tako, da nam omogoča preprosto uporabo na katerikoli uporabljeni platformi. Kot dokaz praktične uporabnosti, smo sistem realizirali na obstoječih napravah ter njegovo delovanje preizkusili tudi v praksi.

V okviru diplomskega dela se bomo najprej seznanili z uporabljenimi tehnologijami ter tehnikami, ki smo jih uporabili za izdelavo končnega produkta. Pri razvoju smo se odločili uporabiti predvsem odprtokodne tehnologije, ki nas ne omejujejo pri distribuciji produkta našim uporabnikom. Sledilo bo poglavje z opisom abstraktnih gradnikov sistema, ki nam omogočajo da lahko na strani strežnika ustvarimo profile odjemalcev ter priklopljenih naprav. Med drugim bomo podrobneje opisali tudi gradnik, ki omogoča združevanja omenjenih profilov v skupine ter gradnik, ki skrbi za samodejne aktivacije naprav. V nadaljevanju se bomo seznanili z delovanjem spletne aplikacije ter kako smo omenjene gradnike uporabili v praksi. Vsakemu opisu so priložene tudi infografike, ki pripomorejo k lažji predstavi izgleda. V zadnjem poglavju si bomo lahko ogledali delovanje odjemalne aplikacije ter postopek glasovnega upravljanja. Med drugim si bomo lahko ogledali, kako smo uporabili postopek razpoznave ter sinteze govora za izboljšanje uporabniške izkušnje ter uporabo storitve 'Wolfram Alpha' [18] za iskanje odgovorov na uporabnikova vprašanja. Naše delo bomo zaključili z diskusijo, v kateri bomo bralca seznanili z rezultati ter težavami, na katere smo naleteli ob razvoju. Tu bomo navedli tudi nekaj smernic bodočega razvoja, ki se jih bomo v prihodnosti skušali držati.

Poglavje 2

Zasnova avtomatizacijskega sistema

V tem poglavju podrobneje opisujemo ključne tehnologije in tehnike, ki smo jih uporabili pri izdelavi sistema. Najprej se bomo seznanili z uporabljenimi vrstami podatkovnih baz ter nadaljevali z opisom programskih ogrodij. Temu bo sledilo podpoglavje z uporabljenimi varnostnimi tehnikami, s katerimi smo izboljšali zaščito našega sistema. Za konec si bomo pogledali še obstoječe možnosti za namestitev ter gostovanje našega sistema.

2.1 Podatkovne baze

Podatkovno bazo lahko opišemo kot organizirano zbirko med seboj pomensko povezanih podatkov, ki je načrtovano z namenom zadovoljevanja informacijskih potreb organizacije. V sistemu smo podatkovne baze uporabili za shranjevanje podatkov, ki so bili vneseni s strani uporabnikov ali zajeti preko priključenih senzorjev. Te lahko nato po potrebi beremo iz nje in jih uporabimo za prikaz informacij ali kot podlago pri sprejemanju nadaljnjih odločitev. Glede na dejstvo, da se potrebe odjemalne ter strežniške aplikacije med seboj razlikujejo, smo se na vsaki strani odločili za drugačno vrsto podatkovne baze. V nadaljevanju poglavja si bomo tako za vsako stran ogledali

našo izbiro ter prednosti, ki jih prinaša v primerjavi z ostalimi.

2.1.1 Strežnik

Podatkovna baza na strežniku se uporablja kot centralizirana podatkovna shramba, ki poleg lastnih podatkov, shranjuje tudi podatke, poslane s posameznih odjemalcev. Ker na strežniku shranjujemo predvsem relacijsko povezane podatke, smo se odločili za uporabo PostgreSQL [6] podatkovne baze. Pri izbiri smo kot glavno primerjavo upoštevali podatkovno bazo tipa MySQL [7]. Glavne prednosti naše izbire smo našteli v naslednjih postavkah:

- odprtokodna ter napredna rešitev,
- objektna usmerjenost,
- hitrejša obravnava kompleksnih procedur,
- zanesljivost ter integriteta podatkov.

PostgreSQL nam med drugim nudi tudi izjemno učinkovitost pri upravljanju večjega števila sočasnih poizvedb, kar pride prav v primeru večjega števila hkratnih dostopov do strežnika. V primerjavi z MySQL podatkovno bazo, nam ta omogoča uporabo dodatnih tipov polj (npr. JSON [9], XML [24]), v katere lahko shranimo JSON ter XML dokumente.

2.1.2 Odjemalec

Podatkovno bazo na odjemalcu uporabljamo za hranjenje lokalnih podatkov, ki jih nato pošiljamo v obdelavo spletni aplikaciji na strežniku. Ob dejstvu, da na odjemalcu ne uporabljamo načela sočasnega dostopa, smo se odločili za podatkovno bazo vrste SQLLite [23] . Glavne prednosti naše izbire smo predstavili v naslednjih postavkah:

- prenosljivost,
- brez namestitve,

- hitro delovanje zaradi podatkovne baze v obliki datoteke,
- idealna za testiranje.

2.2 Ogrodje Bootstrap

Bootstrap [4] je brezplačno ter odprtokodno ogrodje, ki združuje predpripravljene predloge za hitrejše ustvarjanje novih spletnih strani in aplikacij. Predloge so spisane v HTML jeziku ter oblikovane s pomočjo CSS oblikovnih pravil. Gradnike lahko uporabimo za hitro dodajanje tipografskih elementov, gumbov, navigacijskih razdelkov, obrazcev ter preostalih vmesniških komponent na spletno stran. Dodane predloge se izvajajo le na prednjem delu ter so popolnoma neodvisne od zalednega dela aplikacije, ki jo uporabnik razvija. Pri razvoju smo uporabili Bootstrap knjižnico v obliki LESS datotek, ki smo jih ob koncu razvoja s pomočjo CSS predprocesorjev prevedli v CSS obliko. Uporaba LESS pravil nam v primerjavi s CSS pravili omogoča uporabo določenenih programerskih tehnik, kot so npr. zanke, spremenljivke, funkcije ali gnezdenje oblikovnih pravil. Za prevajanje LESS v CSS obliko smo uporabili orodje Gulp, ki omogoča avtomatizacijo ponavljajočih se opravil. Tako smo mu določili, naj ob vsaki opravljeni spremembi nad LESS datotekami, te s pomočjo predprocesorja Gulp-LESS prevede v CSS obliko. Bootstrap ogrodje je bilo uporabljeno predvsem pri izdelavi grafičnega vmesnika spletne aplikacije.

2.3 Ogrodje Django

Django []2 je odprtokodno ogrodje za izdelavo spletnih aplikacij (poglavje 5), spisan s pomočjo programskega jezika Python, ki sledi MVC strukturi aplikacije. MVC je arhitektura razvoja aplikacij, ki temelji na ločenih ravneh. Črka M pomeni pomeni Model, V je View in C pomeni Controller. V osnovi se model uporablja za delo s podatki v podatkovni bazi, kar med drugim vključuje operacije pridobivanja, brisanja ali posodabljanja podatkov. Views

oziroma pogledi, predstavljajo sprednji del spletne aplikacije, kjer so prikazani podatki iz zaledja. Ti so pridobljeni s pomočjo krmilnika (Controller), ki skrbi za upravljanje toka dogodkov glede na prispele zahteve in podatke. Tako ob vsaki prispeli zahtevi iz modelov pridobi ustrezne podatke, jih ustrezno pripravi in posreduje pogledu v prikaz. Ogrodje se uporablja za hitri razvoj kompleksnih spletnih aplikacij, ki hkrati nudi močno podporo delu s podatkovnimi bazami. Pri razvoju Django aplikacij je poudarek predvsem na načelu, da se ne ponavljamo za seboj ter skušamo ponovno uporabiti že spisane funkcionalnosti.

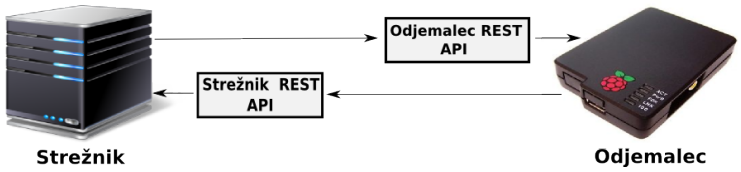
Strežnik: Celotno aplikacijo smo zasnovali na uporabi Django ogrodja ter dodatnih razširitvah Django REST framework [3] (poglavje 2.4) ter Celery [5] (poglavje 2.5) .

Odjemalec: Django je bil skupaj z Django REST framework razširitvijo uporabljen predvsem pri izdelavi storitve, ki se uporablja za sporazumevanje s strežnikom (poglavje 2.4).

2.4 Postopek izmenjave zahtev

Med odjemalno aplikacijo in strežnikom vseskozi poteka izmenjava zahtev, s katerimi si sporočata sveže informacije o stanju. Pri tem je bilo treba poiskati način, s čimer bi obema stranema omogočili, da lahko zahteve sprejemata ter tudi odgovorita na njih. Odločili smo se, da na vsaki strani strani razvijemo storitev, ki bo v zaledju obravnavala prejete zahteve, jih posredovala poslovni logiki ter vrnila ustrezni odziv. Ker na obeh straneh uporabljamo Django ogrodje, smo se odločili za uporabo Django REST Framework razširitve [3], ki je primarno namenjeno razvoju spletnih storitev. Podrobnejši postopek izmenjave zahtev smo opisali v naslednjih korakih.

Priprava zahteve: Obe spletni storitvi podpirata GET, PATCH, POST,



Slika 2.1: Prikaz izmenjave zahteve med strežnikom in odjemalcem

DELETE tipe zahtev. Pri nekaterih je obvezno priložiti nekatere dodatne podatke, ki vsebujejo nove informacije o obnašanju prejemnika. Zaradi lažje izmenjave, smo jih pred pošiljanjem pretvorili v zapis JSON. Ta se nato še dodatno šifrira in podpiše, s čimer lahko prejemniku zagotovimo njihovo verodostojnost (poglavje 2.7.2).

Sprejem zahteve Prejemnik prejme zahtevo ter s pomočjo podpisa preveri pristnost pošiljatelja. V nadaljevanju preveri, ali so bili podatki med pošiljanjem spremenjeni ter nato preda zaledni logiki prejemnika v postopek dešifriranja, ki jih bo povrnila v izvorno obliko.

Obravnava zahteve: Glede na tip prejete zahteve se nato izvedejo ustrezne operacije na prejemniku. V primeru, da smo prejeli zahtevo tipa GET se opravi določena poizvedba nad podatki, katere rezultat se vrne pošiljatelju. Z uporabo DELETE zahteve, lahko zahtevamo nepreklicen izbris določenega zapisa objekta na prejemniku. Izbrisani podatki so nepreklicno odstranjeni ter jih lahko povrnemo v staro obliko le z novo POST zahtevo, ki vsebuje identične zapise ter identifikator kot izbrisani objekt. Če želimo obstoječe podatke le delno posodobiti, lahko uporabimo zahtevo tipa PATCH. Po opravljeni zahtevi se obstoječi zapis posodobi z novimi podatki, ki so bili priloženi zahtevi. Primer praktične uporabe so aktivacije modulov, kjer prejemniku sporočimo novo stanje modula.

Potrditev zahteve: Kot že omenjeno, se kot potrditev ob GET zahtevi vrne seznam objektov, v formatu JSON. V drugih primerih se vrne

status opravljene operacije.

2.5 Način uvrščanja dogodkov

Dogodke (poglavje 4.4) uporabljamo za načrtovane samodejne aktivacije/deaktivacije izbranih modulov ob določenem časovnem terminu. Pri razvoju smo uporabili razširitveni paket Celery [5], s katerim smo v naše Django okolje vključili podporo asinhronemu izvajanju opravil. S praktičnega vidika to pomeni, da se vsa opravila izvajajo ločeno od glavne niti aplikacije. S tem lahko aplikaciji zagotovimo, da ta ne bo nikoli v čakanju na konec izvajanja nekega opravila. Prednosti izbire so naslednje.

- popolna integracija s strani Django ogrodja,
- asinhrono izvajanje,
- izvajanje v realnem času,
- širša podpora s strani razvijalske skupnosti.

Celery nam omogoča izgradnjo asinhrone vrste opravil, katere elementi se nato obravnavajo ter izvajajo v zapisanem vrstnem redu. Vsakemu opravilu je možno določiti periodično izvajanje oziroma mu določiti, da se strogo izvede samo enkrat.

2.6 Prevodi

Pri zasnovi projekta smo se držali priporočenih smernic, da pri uporabi projekta nismo omejeni le na eno jezikovno skupino. Django podpira dodatne funkcionalnosti, s katerimi lahko naš sistem z uporabo prevodov internacionaliziramo ter prilagodimo glede na jezik, ki je bil izbran s strani uporabnika. V trenutni fazi sta podprta jezika slovenščina ter angleščina. Izbira jezika

se izvede na prijavni strani, kjer uporabnik pred prijavo izbere jezik, ki se naj uporabi pri prikazu vsebine. V primeru, da izbira ni opravljena, se kot privzeti jezik uporabi angleščina.

Slika 2.2: Prikaz prijavnih oken v obeh jezikih

2.7 Varnost

V tem poglavju se osredotočamo na opis uporabljenih tehnik, s katerimi smo izboljšali zaščito našega sistema. Zanj je zelo pomembno, da deluje stabilno ter s čim manj izpadi, ki so posledica vdorov v sistem. Da bi lahko to zagotovili, smo se na strani odjemalne naprave in strežnika odločili uvesti nekatere dodatne varnostne tehnike. V naslednjih dveh podpoglavjih se bomo z njimi podrobneje seznanili ter si ogledali, kako so pripomogla k varnosti.

2.7.1 Varna povezava HTTPS

Vsa komunikacija med strežnikom ter odjemalci poteka v obliki pošiljanja HTTP [10] zahtev. Z vidika varnosti je zato pomembno, da morebitnim napadalcem preprečimo prestrezanje poslanih podatkov, s katerimi bi lahko škodovali delovanju sistema. Za zagotovitev varnostnega standarda smo se odločili, da med varnostne zahteve uvrstimo obvezno uvedbo protokola HTPPS [10]. Z njegovo uporabo lahko zagotovimo šifrirano varno povezavo

med pošiljateljem ter prejemnikom podatkov. Kot temelj varnosti se uporablja SSL/TLS enkripcija poslanih podatkov, s čimer imamo zagotovilo o avtentičnosti pošiljatelja. Protokol HTTPS je priporočljivo uvesti na vseh odjemalcih in strežniku. V primeru, da smo za gostovanje uporabili spletno platformo Heroku [8] (poglavje 2.8), je varna povezava že vključena.

2.7.2 Zaščita poslanih podatkov

Temeljno pravilo varnosti se glasi, naj ne zaupamo podatkom, ki so bili poslani s strani nepreverjenih pošiljateljev. Kot dodatno raven varnosti smo se odločili, da vse informacije pred pošiljanjem šifriramo ter podpišemo s strani pošiljatelja. Za vključitev metod kriptografije v naš projekt, smo uporabili že obstoječe funkcionalnosti iz ogrodja Django. Celotni postopek zaščite smo opisali v naslednjih točkah.

Priprava: Vse podatke je potrebno pred pošiljanjem pretvoriti v JSON zapis.

Šifriranje: Vsebino se šifrira s pomočjo kriptografskega algoritma PBKDF2 [11]) ter dodatno 'soljo'. Šifriranje temelji na uporabi 32-mestnega ključa, naključno ustvarjenega ob začetku projekta. Z varnostnih vidikov ga je potrebno skrbno varovati, saj bi vsakršno javno razkritje lahko vodilo v lastnoročno šifrirane ter podpisane podatke s strani napadalcev. V primeru razkritja je ključ možno spremeniti, toda to lahko vodi v neskladnosti s shranjenimi šifriranimi podatki.

Podpis: Šifrirane podatke podpišemo s strani pošiljatelja. S tem postopkom zagotavljamo našo identiteto ter avtentičnost podatkov.

Potrditev: V primeru, da v postopku ni prišlo do zapletov, so podatki sedaj zavarovani ter pripravljeni za pošiljanje. Za dostop do prvotnih podatkov mora prejemnik ponoviti postopek v obratni smeri.

2.8 Gostovanje spletne aplikacije

Kot že omenjeno je naš projekt zasnovan na uporabi osrednje aplikacije na strežniku, ki nadzira in upravlja povezane odjemalce. Za stabilno delovanje sistema je pomembno, da je strežnik vedno dosegljiv ter z čim manj povezavnimi težavami. V primeru, da želimo gostovanje urediti lokalno, je aplikacijo obvezno treba namestiti ločeno od odjemalcev. Načeloma bi lahko ta bila gostovana na katerem od odjemalcev, toda tovrstna praksa z vidika varnosti ni priporočljiva. V primeru, da ne želimo, da je gostovanje aplikacije lokalno, smo za primarnega ponudnika gostovanja podprli spletno platformo Heroku. Ta nam omogoča brezplačno gostovanje v namene razvoja ter testiranja, pri čemer se zaračunavanje storitev začne šele ob povečanem dostopu do aplikacije. Širok nabor vključuje storitve za gostovanje podatkovne baze, pošiljanje elektronske pošte ali izvajanja asinhronih nalog, ki jih uporabljamo tudi mi v našem projektu. Z gostovanjem nismo omejeni le na platformo Heroku, temveč bi lahko gostovali tudi na Amazon EC2, Google App Engine ali katerem izmed manj znanih ponudnikov.

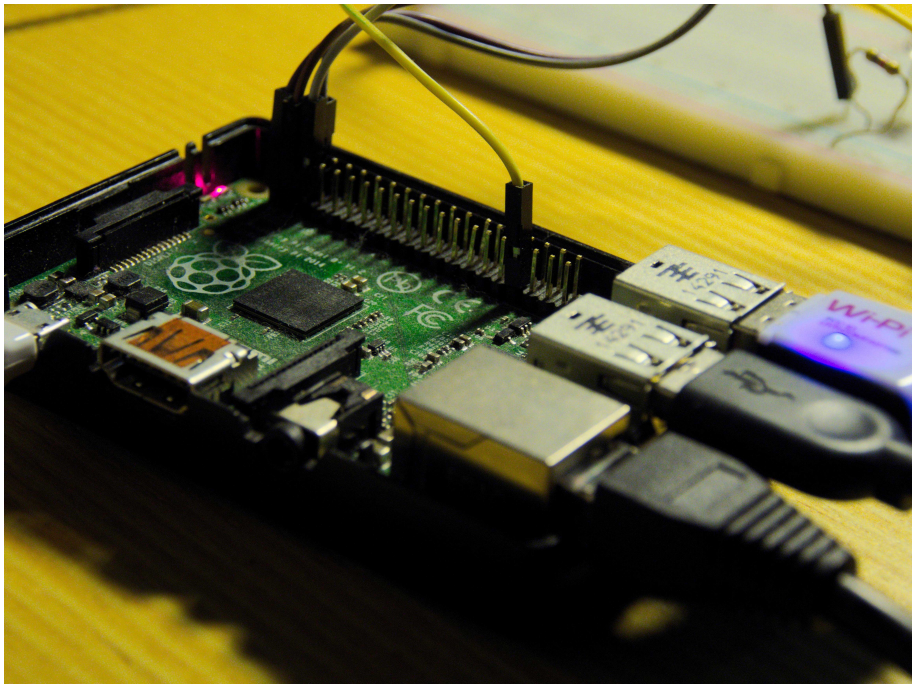
Poglavje 3

Aparaturni gradniki in programska orodja

V poglavju opisujemo naprave ter programska orodja, ki smo jih uporabili pri izdelavi projekta. Praktični primer, kako smo uporabljene aparaturne gradnike povezali v praksi, si lahko ogledamo na sliki 3.2. Pri izbiri orodij smo se osredotočili predvsem na tista, ki so v osnovi brezplačna ter odprtokodna.

3.1 Raspberry Pi B+

Raspberry Pi model B+ [12] je mini računalnik, ki ga lahko glede velikosti primerjamo z bančno kartico. Zasnovan je bil z namenom spodbujanja učenja računalništva v šolah in državah v razvoju, vendar je zaradi svoje nizke cene in majhne porabe energije hitro postal širše priljubljen. Na splošno je najpogosteje uporabljen kot večpredstavnostni predvajalnik, WLAN točko, strežnik ali samo kot poceni nadomestek zmogljivejšega hišnega računalnika. V našem projektu smo se odločili, da Raspberry pi računalnik uporabimo za gostovanje odjemalne aplikacije. Na njem imamo nameščen odprtokodni operacijski sistem Raspbian, ki je prilagojena različica linux distribucije Debian. Za hitrejše delovanje smo prednastavljeno tovarniško hitrost procesorja dvignili s 700 MHz na 900 MHz, s čimer smo hitrost obdelave podatkov teo-

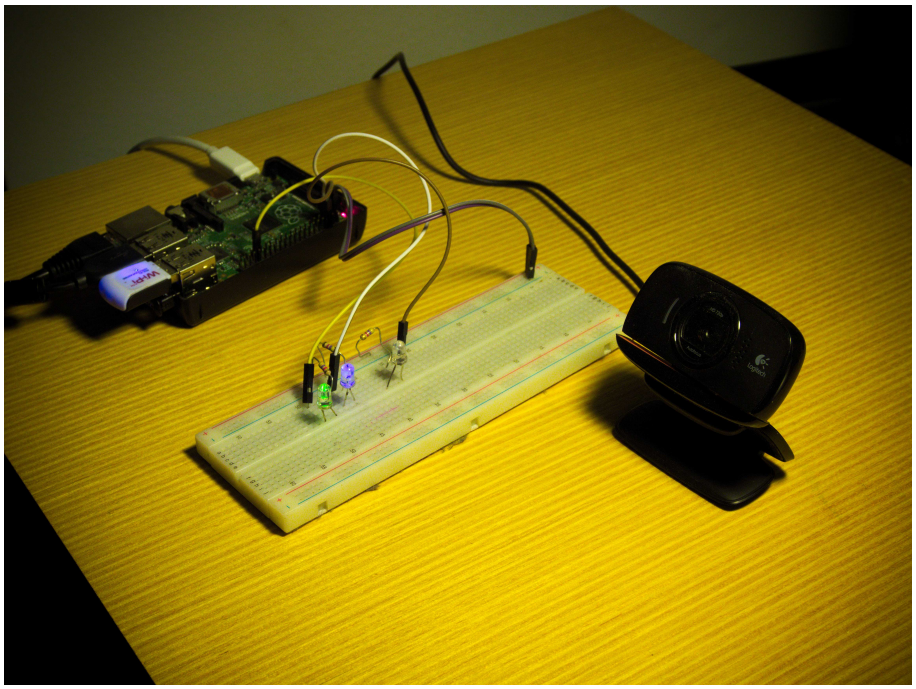


Slika 3.1: Raspberry Pi računalnik

retično zvišali za približno 28.5%. Vezje računalnika vsebuje določeno število integriranih GPIO izhodov, na katere je mogoče priklopiti razne senzorje ali naprave ter jih programsko upravljati. V okviru projekta smo nanj povezane komponente poimenovali z izrazom moduli, ki smo jih podrobneje opisali v nadaljevanju (poglavje 4.1).

3.2 Spletna kamera Logitech C525

V okviru projekta smo se odločili, da kot primarno izbiro za gostovanje odjemalne aplikacije priporočimo uporabo Raspberry Pi računalnika. Glavna pomanjkljivost, ki jo lahko opazimo je odsotnost vhoda za priklop mikrofona, ki ga obvezno potrebujemo za zajem govora. Težavo smo rešili z nakupom spletne kamere Logitech C525 z vgrajenim mikrofonom, ki se na napravo priklopi preko standardnega USB priklopa. Po opravljeni namestitvi je pri-



Slika 3.2: Vezava aparaturnih gradnikov

poročljivo, da opravimo kratko umeritev zajema zvoka, s čimer pripomoremo k natančnejši razpoznavi govora. V primeru, da se odločamo za nakup drugega modela spletne kamere, se je pred nakupom smotrno prepričati ali je ta podprta s strani nameščenega operacijskega sistema Raspbian.

3.3 LED diode

Vezje Raspberry Pi računalnika vsebuje napajalne kontakte (3.3 V ali 5 V), ki v kombinaciji z upravljalno nožico, omogočajo direktni priklop manjših porabnikov energije. V postopku razvoja smo se odločili, da za simulacijo obnašanja fizičnih modulov uporabimo LED diode. V kasnejši, resni uporabi sistema bi lahko te nadomestili z večjimi porabniki, ki bi bili na Raspberry Pi priključeni preko namenskih relejev. Vsaka posamezna LED dioda je v sistem priključena preko 3.3 V vira napajanja, kjer smo pri vsaki diodi dodatno

vezali 270 Ω upor za omejitev električnega toka. Primer uporabljenih LED diod lahko vidimo na sliki 6.5

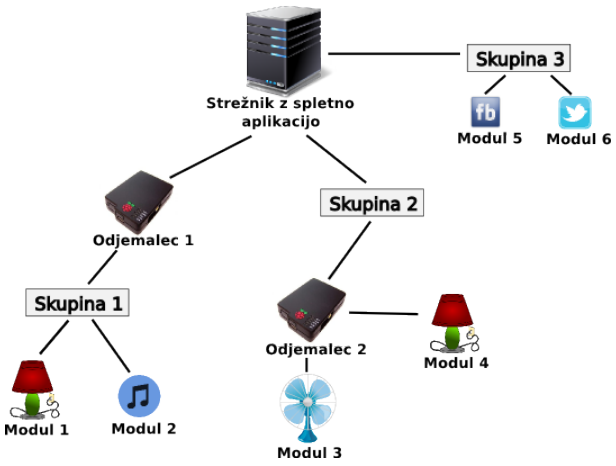
3.4 Urejevalnik Sublime Text Editor

Pri pisanju izvorne kode projekta smo uporabili urejevalnik Sublime Text Editor verzije 3 [13]. Glavno jedro programa je spisano v programskih jezikih C++ ter Python. Namestiti ga je mogoče na več tipov operacijskih sistemov, med katerimi so tudi Windows, Mac OS ter Linux. Privzeto nudi podporo širokemu naboru programskih ter označevalnih jezikov. Te je mogoče razširiti z namestitvijo dodatnih paketov iz spletne skupnosti. Za ponastavitev dela smo namestili dodatne vtičnike za delo s Python, Django, HTMl5, CSS jeziki ter zapisom JSON. Pri razvoju aplikacije za odjemalca smo dodatno namestili še dodatek SFTP, s čimer smo dosegli, da so se vse lokalne spremembe ob shranjevanju avtomatično prenesle na odjemalca.

Poglavje 4

Osnovni konceptualni gradniki

V tem poglavju podrobneje opisujemo ključne gradnike sistema ter njihovo hierarhično povezanost (poglavje 4.1). Njihova uporaba omogoča, da na strani spletne aplikacije ustvarimo profile povezanih odjemalcev, naprav ali pojavov v sistemu. V sklopu slednjih se bomo seznanili z gradnikom, ki omogoča združevanje posameznih modulov v skupine ter spoznali gradnik, ki skrbi za samodejne aktivacije. Ob vsakem se bomo tudi bližje seznanili z njegovim namenom ter pravili, ki določajo njegovo obnašanje.



Slika 4.1: Hierarhija osnovnih gradnikov sistema

4.1 Modul

Modul je hiarhično gledano najnižje umeščen osnovni gradnik sistema, ki mora obvezno imeti dodeljenega odjemalca za svojega starša. Kot modul lahko označimo vsako priklopljeno komponento na odjemalcu ali funkcionalnost sistema, ki upravlja določeno spletno storitev. Z aktivacijo modula smo označili trenutek, ko določeni modul preide iz stanja mirovanja v delovanje. Z drugimi besedami, bi lahko tudi rekli, ko se zgodi vklop neke priključene naprave. Posledično lahko tako vsak modul, ki je v delovanju označimo tudi kot aktiviranega. Obratno smo zaustavitev delovanja označili s pojmom deaktivacija modula. Posamezne module je možno verižno vezati med seboj, nakar se bodo v primeru, da aktiviramo posamezni modul, verižno aktivirali tudi vsi preostali moduli iz verige. Glede na način priklopa ter upravljanja delimo module na naslednja dva tipa:

Fizični: S pojmom označujemo vsako napravo, ki je na odjemalca priklopljena preko integriranega GPIO izhoda. Aktivacije je možno prožiti z glasovno zahtevo, pripravljenim dogodkom ali preko spletnega vmesnika. Vsak izmed načinov aktivacije je pogojen z lastnostjo ali je modul označen kot javno ali zasebno dostopen. Če je modul označen kot zaseben, je aktivacija možna le preko spletnega vmesnika ali z glasovno zahtevo na staršu. Nasprotno so javni moduli dostopni za glasovno aktivacijo z vseh povezanih odjemalcev, ki so povezani v sistem. Primer fizičnega modula lahko vidimo na sliki 6.5.

Fizične module lahko dodatno razvrstimo tudi glede na lastnost ali sistemu posredujejo, ali od njega zahtevajo nove podatke. Tako jih lahko opredelimo kot vhodne ali izhodne fizične module. O vhodnem modulu govorimo takrat, ko je priključena komponenta odgovorna za zbiranje podatkov s pomočjo senzorja. Na podlagi združevanja zajetih podatkov v informacije lahko nato sklepamo nove odločitve o na-

daljnem obnašanju sistema. Med izhodne module lahko uvrstimo vse priključene komponente, katerih stanje delovanja je odvisno od navodil sistema. Klasični primer so stanovanjske luči in gospodinjske naprave. Na sliki 4.1 so fizični moduli označeni kot `modul 1`, `modul 2`, `modul 3` ter `modul 4`.

Abstraktni: Z njimi označujemo vse vgrajene funkcionalnosti sistema, ki so zasnovane za upravljanje s spletnimi storitvami. Kot primer abstraktnega modula lahko navedemo vse funkcije, ki upravljajo storitve, kot so Facebook, Twitter ter Gmail. Vsakemu modulu je mogoče v nastavitvah določiti dejanje, ki ga naj izvede ob začetku aktivacije. Tako lahko med drugim določimo, da se ob aktivaciji izvede poizvedovanje za novo elektronsko pošto, objavami na družabnem omrežju ali sami določimo kratko sporočilo, ki se naj objavi na družabnem omrežju. Na sliki 4.1 lahko vidimo abstraktna modula označena kot `modul 5` ter `modul 6`.

4.2 Odjemalec

Odjemalec je osnovni gradnik sistema; z njim označujemo fizično napravo, na kateri je nameščena odjemalna aplikacija. Vsak odjemalec je hierarhično gledano starš določenemu številu modulov, ki se mu jih določi preko spletnega vmesnika. Ob dodelitvi tako pooblastimo odjemalca, da prevzame odgovornost za njihovo upravljanje, glede na prejete zahteve s strežnika ali glasovnih ukazov. Za nemoteno prejemanje ter pošiljanje zahtev je pomembno, da odjemalcu zagotovimo vzpostavljeno internetno povezavo s strežnikom, preko katere poteka vsa izmenjava zahtev ter podatkov. Za lažje usmerjanje zahtev se na strežniku v nastavitvah odjemalca hranita njegov IP naslov ter morebitna številka vrat, ki se uporabljata za dostop do odjemalne aplikacije na odjemalcu. V primeru, da za odjemalca uporabimo Raspberry Pi računalnik (poglavje 3.1), se vse zunanje naprave nanj priklopijo preko integriranih izho-

dov. Odjemalno aplikacijo je sicer mogoče namestiti tudi na druge naprave, ki podpirajo katero izmed Ubuntu ali Debian Linux distribucij, vendar mu v tem primeru ne moremo določati uporabe fizičnih modulov. Podrobnejša načela delovanja odjemalne aplikacije bomo spoznali v naslednjih poglavjih. Na sliki 4.1 sta odjemalca označena kot `Odjemalec 1` ter `Odjemalec 2`.

4.3 Skupina

Skupina je osnovni gradnik sistema, ki združuje neomejeno število modulov ali odjemalcev v skupino. To je mogoče uporabiti kot bližnjico pri dodajanju elementov v dogodek, s čimer se izognemo ponavljajočemu se dodajanju posameznih elementov. Tu velja pravilo, da prazne skupine ne moremo uvrstiti v dogodek. Ob začetku dogodka se vsi vsebovani elementi skupine hkrati aktivirajo, kar smo poimenovali tudi z izrazom skupinska aktivacija. Če je bil v skupino dodan kateri izmed odjemalcev se posledično aktivirajo vsi moduli, ki jim je ta starš. Skupino je moč vključiti v več različnih dogodkov hkrati ter celo v dogodke, ki se delno ali v celoti časovno prekrivajo.

Izračun aktivacijskega časa v primeru prekrivanj dogodkov z isto skupino lahko razložimo na primeru dveh dogodkov A in B. Če je interval delovanja dogodka B znotraj intervala dogodka A se za skupni čas delovanja skupine izbere dogodek A. V primeru delnega prekrivanja, se kot začetni čas privzame začetek prvega izmed dogodkov medtem, ko je končni čas enak zaključku zadnjega. Če sta oba časovna intervala enaka, ima prednost za aktiviranje skupine tisti dogodek, ki je bil ustvarjen prej. S preračunavanjem aktivacijskega časa se izognemo vmesnemu izklopu modulov ob prehodu med dogodki. Na sliki 4.1 so skupine označene kot `Skupina 1`, `Skupina 2` ter `Skupina 3`.

4.4 Dogodek

S pojmom dogodek označujemo načrtovano samodejno aktiviranje izbranih modulov ali skupin v časovnem intervalu, določenem s strani sistema ali uporabnikov. Vsak dogodek mora obvezno imeti definiran začetni ter končni čas izvajanja, s katerima omejimo interval delovanja. Obe časovni vrednosti morata biti večji ali enaki trenutnemu času, saj velja omejitev, da dogodka ni mogoče uvrstiti v preteklost. Vsakemu je treba določiti tudi seznam modulov ter skupin, za katere želimo, da ob začetku izvajanja preidejo v delovanje. Glede na način, kako so dogodki ustvarjeni, jih delimo na naslednja dva tipa

Ročni: Z njim označujemo dogodek, ki je bil ročno ustvarjen preko spletnega vmesnika. (poglavje 4.1)

Samodejni: S pojmom označujemo dogodek, ki je bil samodejno ustvarjen s strani sistema. Uporabljamo jih za prilagoditev sistema v primeru, da zajeti podatki s vhodnih fizičnih modulov ali abstraktnih modulov presežejo neko določeno mejno vrednost. Omenjenim modulom se lahko preko spletnega vmesnika določi seznam modulov ter skupin, ki se nato ob presegu mejne vrednosti dodajo v nov dogodek. Tako bi lahko določili, da se z zakasnitvijo 5 minut prižge centralno ogrevanje ob padcu temperature v prostoru pod 10 stopinj.

Več o ustvarjanju novih dogodkov s pomočjo vmesnika bomo spoznali v poglavju, kjer opisujemo spletno aplikacijo (poglavje 5.2.4).

Poglavje 5

Delovanje spletne aplikacije

Spletna aplikacija je središče našega sistema, ki avtomatizirano spremlja delovanje nanj priključenih odjemalcev oziroma modulov. Na podlagi zbranih podatkov se lahko nato samostojno odloča o njihovem nadaljnem delovanju. V primeru, da želimo s sistemom upravljati ročno, lahko to storimo preko namenskega spletnega vmesnika. Na strani spletne aplikacije se uporabljajo tudi vsi osnovni gradniki iz poglavja 4. Glede na dejstvo, da z nekaterimi izmed njih označujemo resnične naprave, se vsa izvedena dejanja v spletnem vmesniku odražajo tudi na napravah. Glede na strukturo lahko aplikacijo delimo na sprednji ter zaledni del. Zaledje aplikacije je zadolženo za

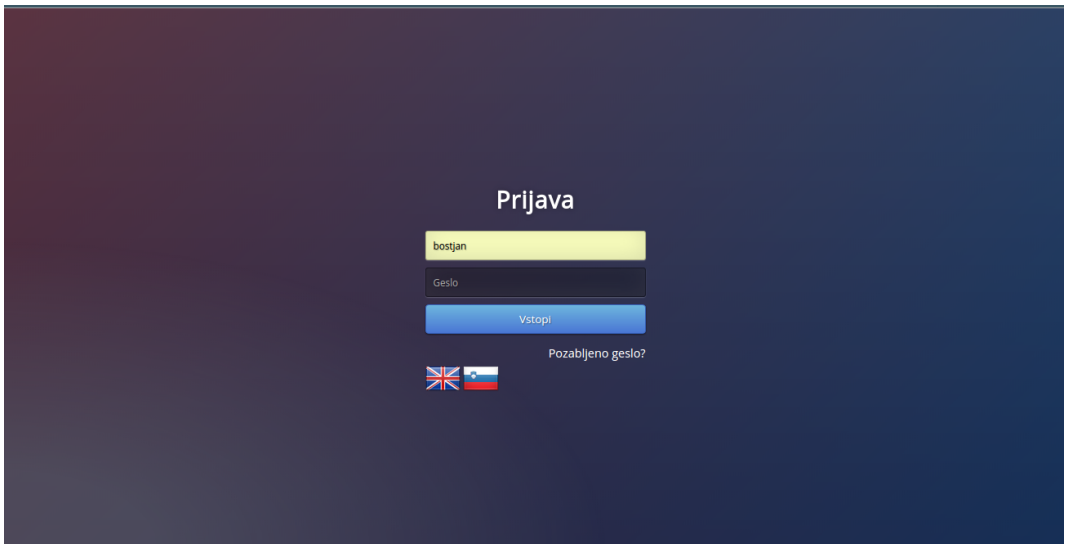
- upravljanje delovanja sistema,
- delu z strežniško podatkovno bazo,
- delu s spletnimi storitvami,
- obdelavo zajetih podatkov.

Tu se nahaja tudi vsa logika, ki določa obnašanje sistema. Za razliko od zaledja nam sprednji del aplikacije predstavlja grafični vmesnik, do katerega dostopamo preko spletnega brskalnika. Uporabniki imajo tako možnost dostopa do sistema s katerekoli naprave, ki omogoča povezavo do svetovnega spleta. Med drugim so tu zbrane informacije o trenutnem stanju sistema,

predstavljene v obliki urejenih preglednic ter grafov. Tu so predstavljeni tudi zajeti podatki s senzorjev, ki so bili poslani v obdelavo s posameznih odjemalcev. Pomembnejša načela delovanja zalednega dela sistema smo podrobneje opisali v poglavju 2, zato smo se v nadaljevanju osredotočili na predstavitev sprednjega dela aplikacije.

5.1 Prijava v sistem

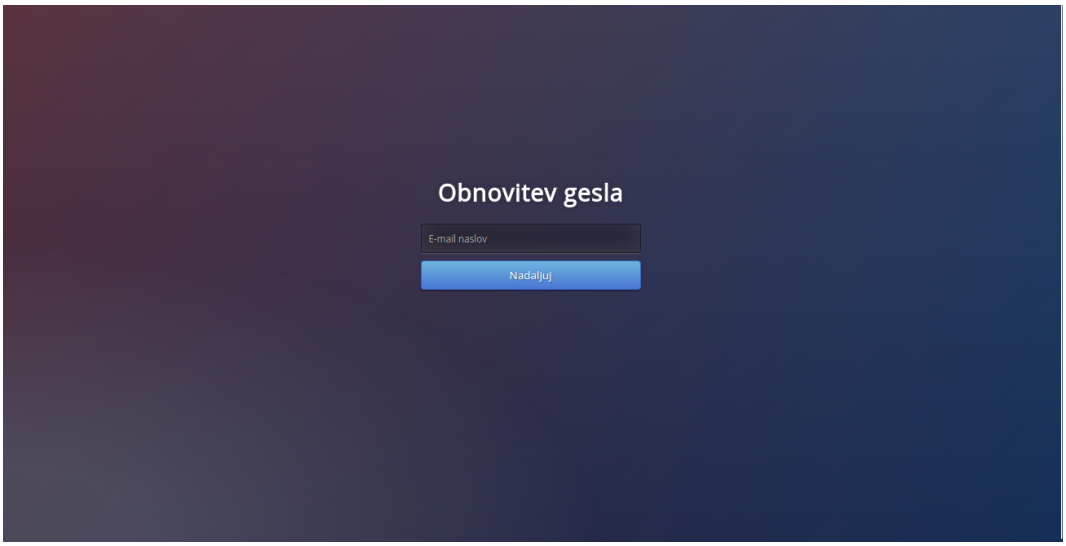
Pri načrtovanju projekta smo se iz varnostnih razlogov odločili, da dostop do spletne aplikacije pogojimo z uspešno prijavo v sistem. Prijavna stran je privzeto mesto, kamor se preusmeri uporabnika v primeru, da ta ni prijavljen ali da mu je potekla prijavna seja. Za uspešno avtorizacijo mora uporabnik tu vpisati svoje uporabniško ime ter pripadajoče geslo, s čimer dokaže svojo identiteto. V primeru, da uporabnik vpiše nepravilno uporabniško ime ali geslo se mu prikaže opozorilo s pozivom za ponovni poskus. Na prijavni strani se nahajajo tudi jezikovne zastavice s pomočjo katerih si uporabnik določi jezik, ki se bo uporabljal pri prikazu strani(poglavje 2.6). V primeru pozabljenega gesla smo na stran umestili povezavo 'Pozabljeno geslo' ob izbiri katere se uporabnika preusmeri na stran za ponastavitev gesla. Podrobnejši postopek obnovitve gesla smo opisali v podpoglavju 5.1.1.



Slika 5.1: Prikaz prijavne spletne strani

5.1.1 Pozabljeno geslo

V primeru pozabljenega uporabniškega gesla je bilo potrebno uporabnikom omogočiti možnost, da le tega obnovijo. Kot rešitev problema smo ločeno od glavne aplikacije dodali spletno mesto, kjer lahko uporabniki z vpisom svojega elektronskega naslova zahtevajo ponastavitev gesla. Ker v sistemu velja omejitev, da mora vsak uporabnik imeti unikatno določen elektronski naslov, je s tem zagotovljeno, da so navodila za obnovitev gesla resnično poslana na pravi naslov. V postopku se uporabniku dodeli začasno naključno generirano geslo, ki se nato priloži v elektronskem sporočilu. Ob uspešni prijavi je priporočljiva zamenjava gesla, saj je začasno geslo veljavno le 14 dni. V primeru, da želimo veljavno dobo začasnega gesla spremeniti lahko to storimo v nastavitvah spletne aplikacije. Primer poslanega elektronskega sporočila lahko vidimo v naslednjem odstavku.



Slika 5.2: Obnovitev pozabljenega gesla

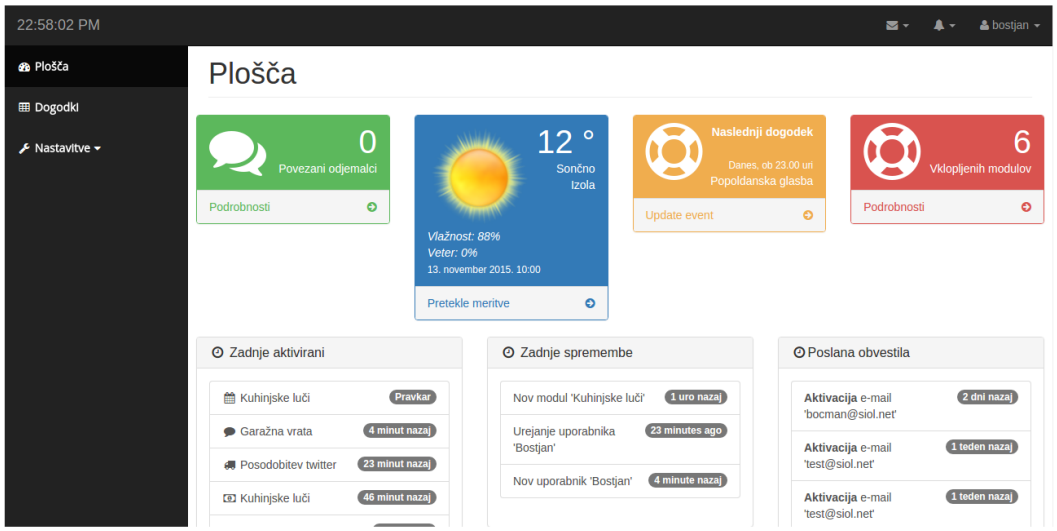
To e-pošto ste prejeli, ker ste zahtevali ponastavitev gesla za vaš uporabniški račun na strani www.example.com.

Vaše začasno geslo je: kioh78e237ddms2sd8dhe32jkds9863
prosimo, da ga spremenite po uspešni prijavi
Vaše uporabniško ime (za vsak primer): testni_uporabnik
Hvala, ker uporabljate našo stran!

Ekipa strani www.example.com

5.2 Plošča

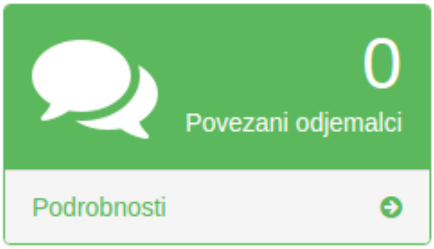
V okviru poglavja si bomo ogledali glavno stran spletne aplikacije, ki v grafično prijazni obliki predstavlja najpomembnejše informacije iz delovanja sistema. Tu si bomo ogledali predvsem najpomembnejše gradnike, iz katerih je sestavljena ter njihove morebitne podstrani.



Slika 5.3: Glavna plošča spletne aplikacije

5.2.1 Dejavni odjemalci

Za lažji pregled nad skupnim številom povezanih odjemalcev smo se odločili, da na glavno stran aplikacije umestimo gradnik, ki bo prikazoval to informacijo. Za pridobitev podatkov smo uporabili AJAX zahtevo [25], ki nam v minutnem časovnem razmiku samodejno osvežuje podatke na gradniku. Ob nastopu zahteve se ta posreduje v zaledje aplikacije, kjer se v podatkovni bazi izvede selekcijska poizvedba. Pri iskanju povezanih odjemalcev se preverja vrednost atributa is_Connected, ki označuje, ali je odjemalec v zadnji minuti posodobil svoje stanje kot dejavno. Skupni seštevek dejavnih odjemalcev se nato vrne spletnemu gradniku, ki poskrbi za osvežitev stanja. Za podrobnejše informacije smo v spodnji del gradnika dodali povezavo "Podrobnosti", ki nas ob izbiri preusmeri na seznam vseh odjemalcev z označenimi dejavnimi odjemalci.

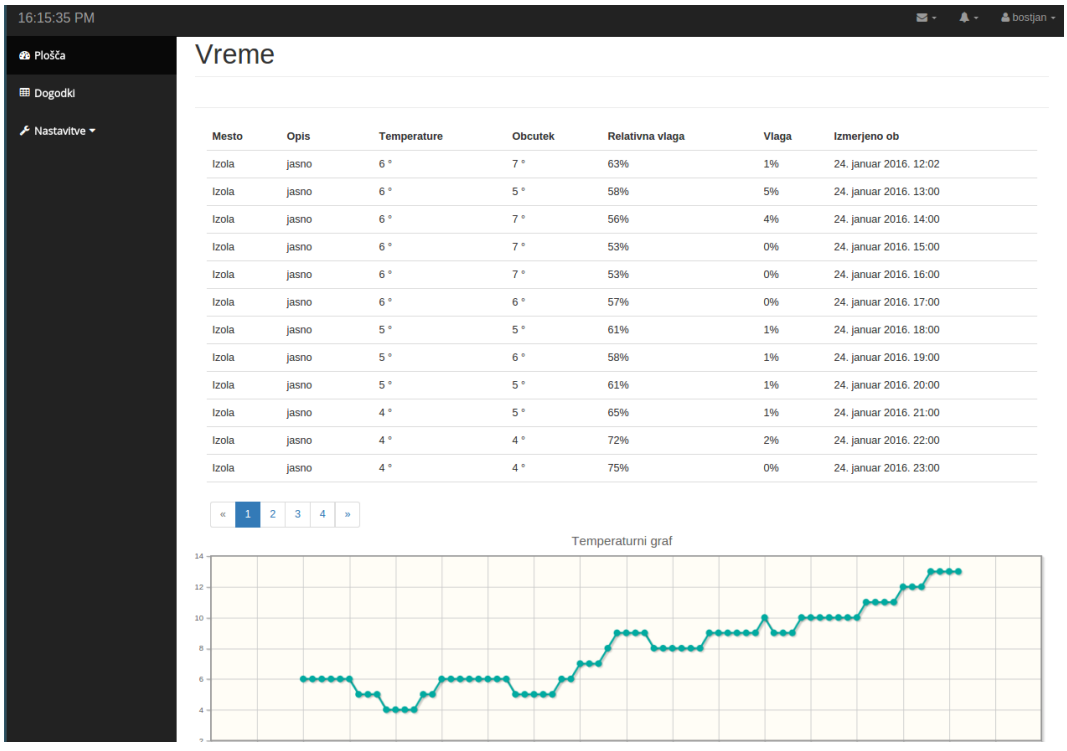


Slika 5.4: Vizualni gradnik za prikaz števila dejavnih odjemalcev

5.2.2 Prikaz vremena

Vizualni gradnik je namenjen prikazu zadnjih vremenskih informacij za kraj, ki se določi v nastavitvah spletne aplikacije. Kot vir vremenskih informacij se uporablja spletna stran 'Wunderground.com' [19], ki omogoča dostop do vremenskih podatkov preko namenske spletne storitve. Ob vsaki poslani zahtevi moramo obvezno priložiti unikatni identifikator, ki smo ga pridobili ob opravljeni registraciji na spletni strani. V primeru, da identifikator ni bil priložen, ali je neveljaven, se zahteva zavrne. Če je bila avtorizacija uspešna se nam, kot rezultat posredujejo vremenski podatki v zapisu JSON. Med prejetimi podatki se nahaja tudi vremenska napoved za naslednjih 6 ur na podlagi katere se lahko pripravi avtomatske dogodke kot odziv na določene vremenske pojave. Kot primer uporabe lahko navedemo zaprtje garažnih vrat ob bližajoči se nevihti ali predvajanje zvočnega obvestila na odjemalcih ob napovedani toči. Za podrobnejši pregled nad zajetimi meritvami smo v spodnji del gradnika umestili povezavo "Pretekle meritve". Ob izbiri povezave je uporabnik preusmerjen na spletno mesto z zbranimi meritvami, ki so predstavljene v obliki preglednice ter grafa.

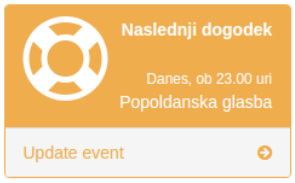
Slika 5.5: Vizualni gradnik za prikaz vremena



Slika 5.6: Celotni prikaz vremenskih meritev

5.2.3 Bližnji dogodek

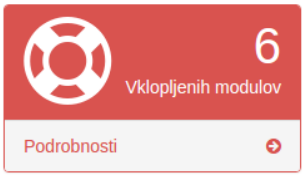
Namen gradnika je prikaz določenih informacij o dogodku, ki je naslednji v izvajalni vrsti. Kot glavno merilo se pri poizvedovanju upošteva začetni čas dogodka, ki mora biti v čim manjšem časovnem razmiku od trenutnega. Pri načrtovanju gradnika smo se odločili, da poleg imena nanj umestimo tudi začetni ter končni čas izvajanja dogodka. S tem imajo uporabniki lažjo predstavo o aktivacijskem času dogodka, kar lahko uporabijo pri nadaljnjih odločitvah. Za samodejno osveževanje podatkov na gradniku smo uporabili AJAX zahtevo, ki v minutnem časovnem razmiku od strežnika zahteva najnovejše informacije o naslednjem dogodku v vrsti. V primeru, da uporabnik želi več informacij smo v spodnji del gradnika umestili povezavo "Več informacij", ki nas ob izbiri preusmeri na profil dogodka.



Slika 5.7: Vizualni gradnik za prikaz naslednjega dogodka

5.2.4 Aktivni moduli

Vizualni gradnik je namenjen prikazu informacije o skupnem številu aktivnih modulov. S tem smo uporabnikom omogočili lažji nadzor nad aktivnimi moduli, ki jih lahko v primeru nezaželjenega delovanja hitreje zaustavijo. Za samodejno osveževanje smo uporabili AJAX zahtevo, ki z minutnim časovnim razmikom posodablja informacije na gradniku. V spodnji del gradnika smo umestili povezavo "Podrobnosti", ki nas ob izbiri preusmeri na spletno mesto vseh aktivnih modulov.



Slika 5.8: Vizualni gradnik za prikaz števila aktivnih modulov

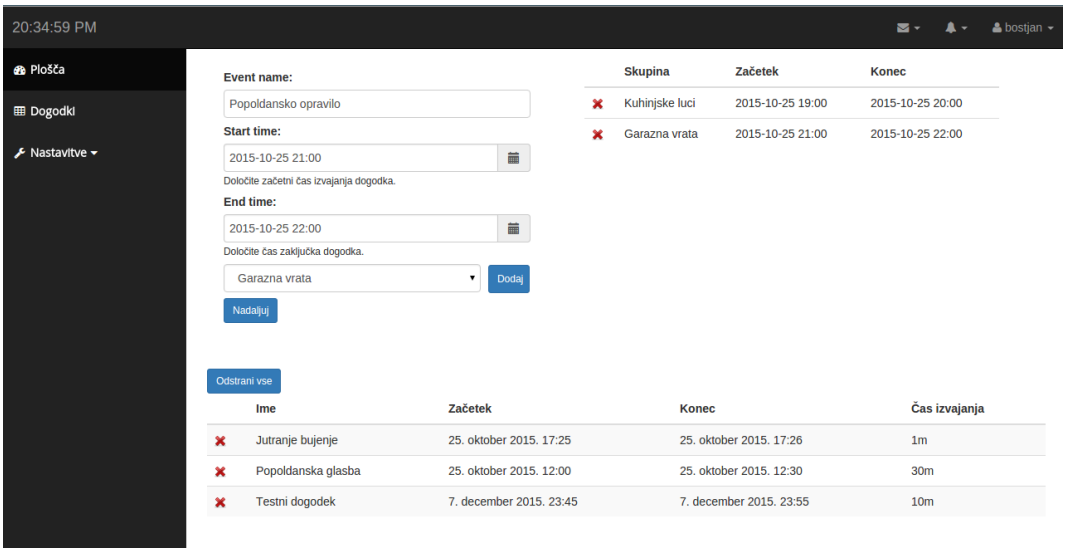
5.3 Dogodki

V našem sistemu uporabljamo dogodke za načrtovano aktivacijo izbranih modulov oziroma odjemalcev v časovnem intervalu, ki je določen s strani uporabnikov. Za lažje delo z dogodki smo v aplikaciji pripravili spletno mesto, kjer se dogodke ustvarja ali ureja s pomočjo vmesnika. Vse opravljene spremembe se ob potrditvi pošljejo v zaledje, kjer se podatke obravnava glede na tip poslane zahteve. Za delo z dogodki smo na strani pripravili naslednja polja

Ime dogodka - Tu se pričakuje vnos imena dogodka po katerem ga bomo lahko ločili od preostalih dogodkov. Ime je lahko poljubno izbrano saj tu nismo omejeni na unikatna imena. Glavni pogoj, ki velja je, da je ime dolgo vsaj 6 znakov pri čemer je dovoljena uporaba vseh črk, števil presledka ali podčrtaja. V primeru, da ime ne ustreza predpisanim merilom se pod poljem prikaže opozorilo, ki nas poziva naj vrednost popravimo.

Začetni čas - V polje se pričakuje vnos časovne vrednosti s katero določimo, kdaj naj se dogodek prične izvajati. Za lažji vnos smo polju dodali izbirnik v obliki koledarja, s katerim lahko določimo časovno vrednost na grafični način. V primeru, da je vnesena vrednost večja od časa zaključka dogodka se pod poljem izpiše ustrezno opozorilo.

Končni čas - Služi vnosu časovne vrednosti ob kateri se bo dogodek prenehal izvajati. Način vnosa je isti kot pri začetnem času, s to razliko, da mora biti vnešena vrednost obvezno večja od začetnega časa.



Slika 5.9: Prikaz dodajanja novega dogodka

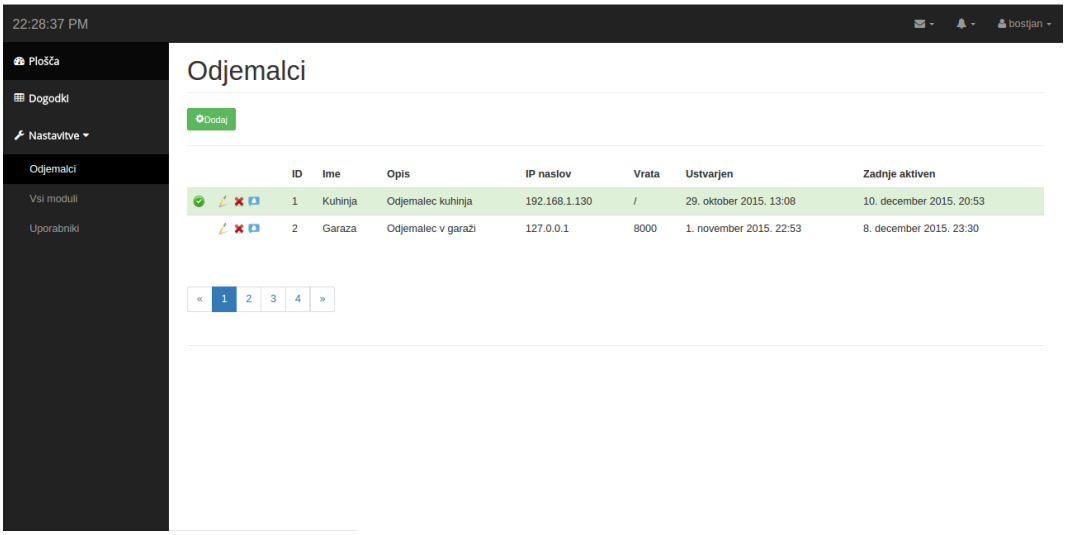
Izbirnik modulov - Izbirnik omogoča izbiro modula, ki naj sodeluje v dogodku iz množice vseh zbranih modulov. Na seznam so uvrščeni vsi dosegljivi moduli, ne glede na to ali so fizičnega ali abstraktnega tipa. V primeru, da trenutno ni dosegljivih fizičnih modulov oziroma, da še ni bil dodan noben abstraktni modul, se ustvarjanje novega dogodka onemogoči. Teoretično bi lahko dovolili ustvarjanje praznih dogodkov, vendar bi bilo to z vidika uporabe nesmiselno.

Ob vsaki spremembi vrednosti v izbirniku se nam pod vnosnimi polji prikažejo vsi trenutno pripravljeni dogodki, ki v svojih aktivacijah uporabljajo izbrani modul. Ob izbirniku se nahaja gumb 'Dodaj' s katerim uporabniki izbrani modul uvrstijo v začasni izvajalni seznam, ki se prikaže na desni strani zaslona (slika 6.3). Ob potrditvi se vse vrednosti iz začasnega seznama prenesejo v zaledje aplikacije, kjer se dodani elementi preslikajo v nove dogodke. V primeru, da je določeni modul že rezerviran, se pod izbirnikom modulov prikaže opozorilo, ki nas poziva naj uredimo moteči modul.

5.4 Nastavitve odjemalcev

V razdelek smo uvrstili vse nastavitve, ki vplivajo na delo z odjemalci. Ob izbiri razdelka se uporabnika preusmeri na spletno mesto, kjer so v obliki preglednice zbrani vsi odjemalci, ki so bili s strani uporabnikov dodani v sistem. Tu so zbrane vse informacije o posameznih odjemalcih ter njihovih najpomembnejših lastnostih. Poleg imena ter kratkega opisa lahko tu vidimo, kdaj je bil ta ustvarjen ter časovni žig, ki označuje njegovo zadnjo aktivnost. Vse informacije o dosegljivosti se osvežujejo v minutnem časovnem koraku, razen če odjemalec eksplicitno obvesti strežnik, da bo prešel v stanje nedosegljivosti.

Na vrhu preglednice se nahaja gumb 'Dodaj', ki nas ob izbiri preusmeri na spletno mesto kjer se v sistem dodaja nove odjemalne profile. V primeru, da je odjemalec trenutno označen kot dosegljiv, se njegova vrstica poudari z zeleno barvo ter utripajočo ikono. Ob vsakem odjemalcu se poleg imena nahajata ikoni, ki označujeta možnost njegovega urejanja ali brisanja. Če izberemo izbris odjemalca je ta v celoti odstranjen iz sistema ter s tem nedosegljiv za prejemanje poslanih zahtev. Če bi želeli z odjemalno napravo znova vzpostaviti povezavo, bi morali ponovno ustvariti odjemalni gradnik z nastavitvami izbrisanega profila.



Slika 5.10: Prikaz seznama vseh odjemalcev

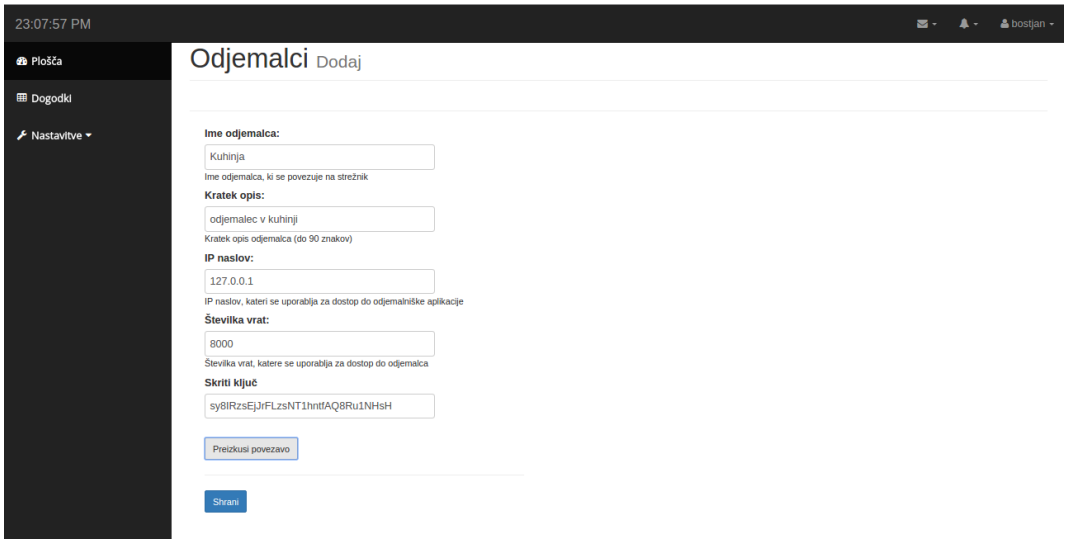
Dodajanje novega odjemalca

Ob vsaki namestitvi odjemalne aplikacije na določeno napravo je potrebno na strani strežnika ustvariti nov odjemalni profil naprave, ki se bo v nadaljevanju uporabljal za sporazumevanje s to napravo. Spletno mesto nam omogoča, da s pomočjo vmesnika izpolnimo vrednosti, katere so zahtevane pri gradniku Odjemalec ter se bodo nato uporabljale pri sporazumevanju z odjemalnimi napravami. Ob potrditvi so podatki poslani v zaledje aplikacije, kjer se ob koncu obravnave preslikajo v nov gradnik Odjemalec. Posledično se bodo vse zahteve, ki jih bomo v nadaljevanju opravljali nad gradnikom neposredno pošiljale tej odjemalni napravi. Na strani se nahajajo naslednja polja.

Ime odjemalca - Se uporablja za označitev odjemalca po katerem ga bomo lahko ločili od preostalih odjemalcev. Obvezno mora biti unikatno ter krajše od 30 znakov.

Kratek opis - Je namenjen kratkemu opisu odjemalca oziroma opisu naloge, kateri je odjemalec primarno namenjen.

IP naslov - Hrani IP naslov odjemalca z nameščeno odjemalno aplikacijo.



Slika 5.11: Prikaz dodajanja novega odjemalca

Na ta naslov se bodo s strežnika usmerjale vse zahteve, za katere želimo, da jih ta specifični odjemalec prejme v obravnavo. Tu je pomembno, da je naslov pravi, saj je drugače sporazumevanje s to napravo neuspešno.

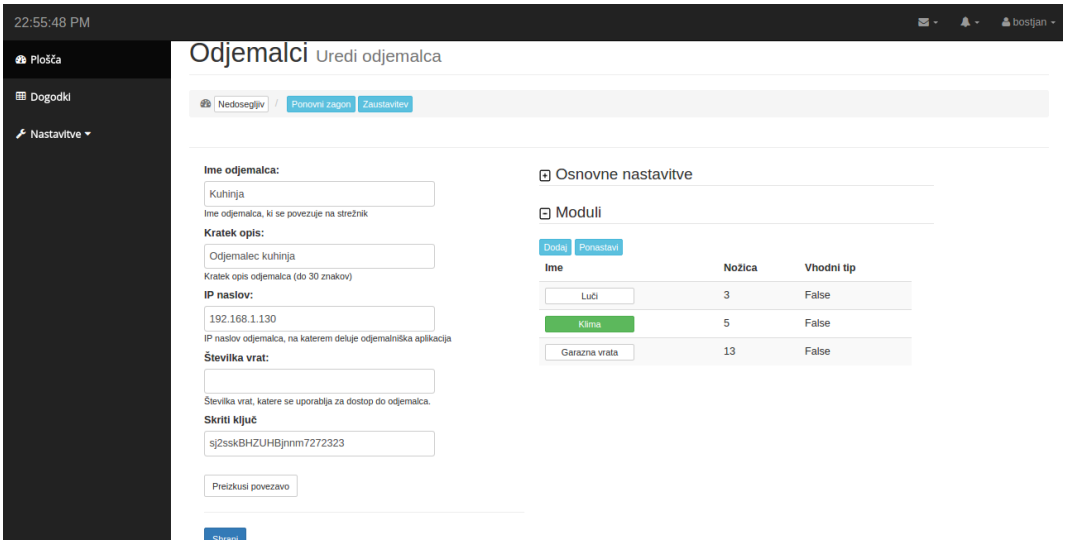
Vrata - V primeru, da se za dostop do odjemalne aplikacije uporabljajo vrata se tu vpiše številko vrat, ki jih ta uporablja. To uporabimo v primeru, kadar so na odjemalcu v izvajanju še drugi procesi, saj s tem dosežemo direktno usmeritev zahtev.

Skriti ključ - Se uporablja kot dodatna zaščita pri ugotavljanju identitete odjemalca, ki je strežniku poslal zahtevo. S tem preprečimo morebitno zlorabo s strani napadalcev, ki bi lahko poneverjali zahteve in s tem vplivali na delovanje sistema. Ključ mora biti obvezno unikaten ter daljši od 10-ih znakov, vsebovati mora vsaj eno števko ter veliko črko.

Urejanje nastavitev odjemalca

Na skupnem seznamu vseh odjemalcev lahko ob vsakemu odjemalcu najdemo ikono za urejanje njegovih nastavitev. Ob izbiri nas spletna aplikacija pre-

usmeri na spletno mesto, kjer se prikažejo njegove nastavitve, katere lahko poljubno urejamo. Prikazana so nam ista polja kot pri ustvarjanju novega odjemalca, s to razliko, da so zdaj zapolnjena z vrednosti iz profila izbranega odjemalca. Glavna razlika se nam prikaže z dodatnimi možnostmi, saj lahko tu najdemo dodatna zavihka z lokalnimi nastavitvami odjemalca ter njegovimi moduli, katere je možno ročno aktivirati. Ob vsakem modulu se nahaja tudi opomba, katera nožica mu je bila dodeljena ter ali je vhodnega ali izhodnega tipa. V primeru, da želimo odjemalec zaradi problematičnega delovanja ročno zaustaviti ali ponovno zagnati, smo zraven imena odjemalca umestili gumba, ki nam to omogočata.

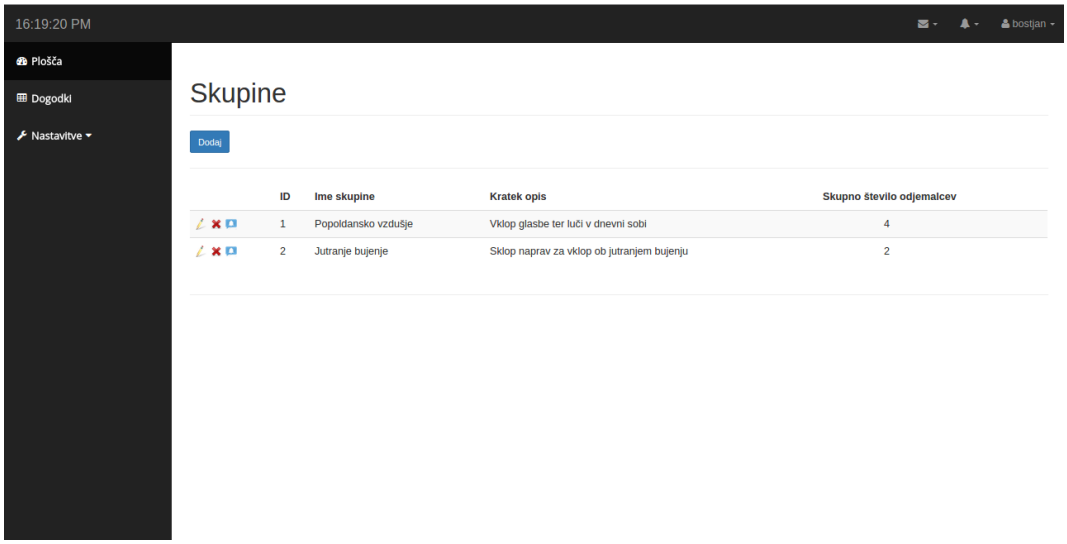


Slika 5.12: Prikaz nastavitev odjemalca

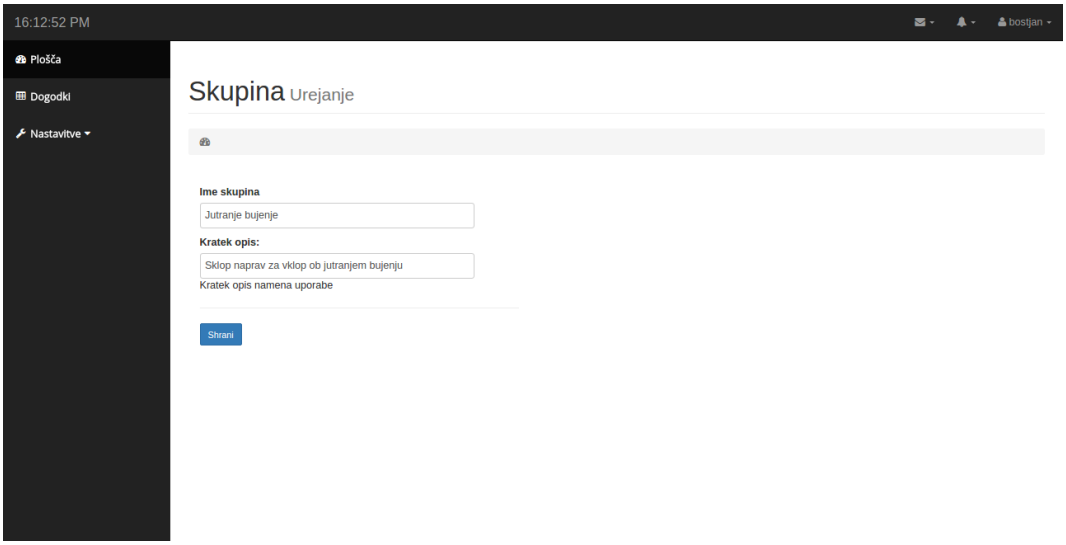
5.4.1 Skupine

V nastavitvah spletne aplikacije se nahaja razdelek Skupine, ki nas ob izbiri preusmeri na stran s pregledom vseh do sedaj dodanih skupin (slika 5.4.1).

Tu so ob vsaki skupini naštete najpomembnejše informacije, kot so ime, opis ter skupno število modulov, ki so dodeljeni tej skupini. Ob vsaki skupini se nahajata tudi možnosti izbrisa ali urejanja te skupine. Če želimo dodati novo skupino, lahko to storimo z izbiro gumba "Dodaj skupino", ki se nahaja v zgornjem delu strani. Tu mora uporabnik obvezno vpisati ime skupine ter njen kratek opis. Ob uspešnem ustvarjanju se ustvari nova skupina, pod pogoji ter pravili, ki so bili opisani v opisu gradnika (poglavje 4.1). Kot lahko vidimo na sliki 5.4.1 ostanejo polja ob urejanju skupine nespremenjena.



Slika 5.13: Prikaz seznama vseh skupin



Slika 5.14: Prikaz urejanja skupine

Poglavje 6

Delovanje odjemalca

V sistemu uporabljamo odjemalce kot naprave, na katere lahko priklapljamo posamezne naprave (fizične module). Te je nato mogoče upravljati preko spletnega vmesnika ali s pomočjo govora. Sama naprava se sama po sebi ne zaveda prisotnosti strežnika ter priključenih modulov, zato je nanjo potrebno namestiti odjemalno aplikacijo, ki bo postala vmesni člen med strojnim delom naprave ter preostalim sistemom. Aplikacija deluje kot samostojna enota, ki vse prispele zahteve posreduje zaledni logiki, ki poskrbi, da se naprava nanje ustrezno odzove. Kot smo že omenili v preteklih poglavjih je podprta možnost glasovnega upravljanja, ki nam omogoča, da s pomočjo govora upravljamo obnašanje sistema. Med drugim smo razvili tudi podporo spletni storitvi Wolfram Alpha, s pomočjo katere lahko z govorom zastavimo aplikaciji vprašanje, ta nam pa poskuša s pomočje storitve poiskati odgovor nanj. Aplikacija je odgovorna tudi za zbiranje svežih podatkov, kateri se zajemajo preko priključenih senzorjev. Ob koncu vsakega zajema, se ti samodejno pošiljajo na strežnik v dodatno obdelavo. Informacije se nato uporabljajo za prikaz uporabnih informacij ali kot podlaga za ustvarjanje samodejnih dogodkov. V primeru, da strežnik ne deluje v lokalnem omrežju je pomembno, da napravi zagotovimo povezavo na svetovni splet, preko katere se lahko sporazumeva s strežnikom ter storitvami, potrebnimi za podporo govornemu upravljanju. Ta se med drugim uporablja za izmenjavo aktivacijskih ozi-

roma deaktivacijskih zahtev, ki določajo novo stanje modulov. V primeru uporabe lokalnega omrežja ta sicer ni obvezni pogoj, vendar je posledično s tem glasovno upravljanje onemogočeno. Ob stanju popolne odsotnosti se vsi podatki z vhodnih fizičnih modulov shranjujejo lokalno ter se ob ponovni dosegljivosti pošljejo z zakasnitvijo na strežnik.

6.1 Namestitvene zahteve

Pred namestitvijo aplikacije je potrebno na napravi spremeniti nekatere odjemalne nastavitve. Potrebne spremenljivke, ki potrebujejo naše urejanje, so naštete v spodnjih postavkah:

Up_ime - Tu je treba vpisati uporabniško ime, ki smo ga pridobili ob ustvarjanju uporabniškega profila na strani spletne aplikacije.

Geslo - Geslo, ki pripada uporabniškemu računu. V primeru, da smo ga pozabili, ga je možno obnoviti preko spletnega vmesnika.

Domena - Hrani spletni naslov do strežnika, kjer deluje spletna aplikacija.

Vrata - V primeru, da se pri dostopu uporabljajo specifična vrata, jih je treba tu vpisati.

Odjemalec_kljuc - Vrednost skritega ključa, ki je bil odjemalcu unikatno določen na strani strežnika.

WolframAlfa_ID - Vrednost ključa, ki ga uporablja Wolfram Alfa storitev pri povpraševanju. Pridobimo ga po opravljeni registraciji na njihovi uradni strani.

Odjemalec_tip - Tip odjemalca, na katerem je nameščena odjemalna aplikacija.

V primeru, da sistem deluje na lokalni ravni, je priporočljivo vsaki napravi določiti svoj lokalni naslov. Sicer se za dostop do upravljanja posameznih naprav uporabljajo unikatna vrata na strani odjemalne aplikacije. S tem spletni aplikaciji zagotovimo, da pri pošiljanju zahtev ne bo prihajalo do težav pri usmerjanju.

6.2 Sinteza govora

Sinteza govora je postopek pretvorbe besedila v umetno tvorjeno obliko, ki poskuša posnemati človeški govor. V sistemu se sinteza uporablja na strani odjemalca, kot eden interaktivnih načinov obveščanja uporabnikov. Uporablja se povsod, kjer želimo neko sporočilo v tekstovni obliki pretvoriti v govor, ki bo nato preko priključenih zvočnikov predvajalo uporabniku. Kot primer uporabe bi lahko odjemalcu določili, da nas redno glasovno obvešča o stanju modula, ki je v bližnji prihodnosti uvrščen v neki dogodek.
Kot merilo pri izbiri primernega načina pretvorbe, smo uporabili naslednja merila

- primerljivost z naravnim govorom,
- podprt s strani Python programskega jezika,
- majhna občutljivost na šume,
- brezplačnost.

Na podlagi primerjanja rezultatov smo se odločili, da uporabimo Google Speech spletno storitev, ki se uporablja tudi pri storitvi Google Translate ter v primerjavi z ostalimi storitvami nudi najboljši približek naravnemu govoru. Za sporazumevanje s spletno storitvijo smo uporabili razširitveni Python paket gTTS [15](Google Text To Speech) s pomočjo katerega smo pripravili lastni razred, ki združuje posamezne metode za sintezo govora. Instanca razreda tako sprejme besedilo kot vhodni parameter ter ga pripravi v zahtevano

Slika 6.1: Postopek sinteze govora

obliko. V postopku se zamenjajo vsi šumnike s sorodnimi črkami ter odstranijo morebitni odvečni presledki. Besedilo se nato z uporabo HTTP zahteve pošlje storitvi v pretvorbo, ki nam nato kot končni rezultat pretvorjeni govor v formatu MP3.

6.3 Storitev WolframAlfa

Wolfram Alfa je spletna storitev, ki svojim uporabnikom omogoča uporabo zmogljivega povpraševalnega pogona. Z njim si lahko pomagamo pri iskanju primernega odgovora na poljubno zastavljeno vprašanje. Ta so lahko v semantični ali matematični obliki, saj nam ta med drugim tudi nudi tudi močno podporo delu z matematični kalkulacijami. Vsa semantična vprašanja morajo obvezno biti zastavljena v angleškem jeziku, saj se ta privzeto uporablja za iskanje informacij. Primer zastavljenega vprašanja bi lahko bil 'What's my IP address' (Kakšen je moj IP naslov) nakar bi kot odgovor prejeli podatke o IP naslovu, ki je določen naši napravi. Iskanje poteka s pomočjo optimiziranih algoritmov, ki nam poiščejo odgovor med številnimi povezanimi podatkovnimi bazami s to tematiko. Pri iskanju se uporabljajo tako akademski kot komercialni viri informacij. Med njih se uvrščajo razni zavodi, knjižnice ter celo Facebookovi računi. Vsi akademski podatki so bili pred objavo dodatno pregledani ter potrjeni s strani poznavalcev stroke, s čimer se lahko zanesemo na resničnost prejetih informacij. Za opravljanje povpraševanj je bilo treba na uradni strani opraviti brezplačno registracijo, s čimer smo pridobili svoj unikatni ključ. Ta se v nadaljevanju uporablja pri dokazovanju naše identitete ob vsaki novi povpraševalni zahtevi. Z brez-

plačnim računom lahko z njim opravimo do 2000 mesečnih nekomercialnih povpraševanj.

6.4 Razpoznava govora

Razpoznava govora je proces, v katerem skušamo s pomočjo podpore računalnika razpoznati človeški govor. Že ob začetku 90-ih let prejšnjega stoletja so se postopoma razvijali sistemi, ki so v omejenem obsegu znali razpoznati besede iz omejene množice besed. V tem času je na trg prišlo že dosti razpoznavnih pogonov z veliko boljšo kakovostjo razpoznave, vendar zaradi nekomercialne uporabe niso postali širše uporabljeni. V trenutnem času razvoja se položaj spreminja, saj je področje postalo ponovno zanimivo z vključitvijo le teh v pametne naprave ter operacijske sisteme. Pri izbiri storitve za razpoznavo govora smo uporabili naslednje merila:

- stopnja uspešne razpoznave,
- sposobnost razpoznave brez učne množice,
- točnost razpoznave, kljub prisotnim motnjam,
- podprtost s strani Python programskega jezika,
- brezplačnost.

Glede na rezultate primerjanja različnih tehnologij smo se odločili, da uporabimo spletno storitev Google Speech Recognition, ki je deloma vključena tudi v spletni storitvi Google Translate. Kot primarni jezik razpoznave smo uporabili angleščino, saj je bila uspešnost razpoznave v primerjavi z drugimi naravnimi jeziki neprimerno boljša. Postopek razpoznave smo opisali v naslednjih postavkah:

Stanje pripravljenosti - Instanca razreda za razpoznavo govora se neprekinjeno izvaja v zaledju aplikacije ter poskuša preko priključenega mikrofona zaznati zvočne spremembe v prostoru. Če aplikacija zazna

Slika 6.2: Postopek razpoznave govora

nenadne zvočne spremembe v prostoru, jih ta takoj začne zajemati v nestisnjenem formatu WAV. Snemanje se zaključi šele, ko govoru sledi določeni interval tišine, ki ga je možno poljubno spreminjati v nastavitvah odjemalne naprave (privzeto 2 sekundi). Pri razpoznavi govora smo naleteli na težave, če smo imeli vklopljeno razpoznavo govora v času predvajanja sintetiziranega govora. Odjemalna aplikacija je namreč zaznala zvočne spremembe ter poskušala izvesti razpoznavo le tega. Posledično se je postopek začel ciklično ponavljati in s tem onemogočil normalno razpoznavo. Problem smo rešili z začasno zaustavitvijo procesa razpoznave govora v času predvajanja posnetka.

Priprava zapisa - Spletna storitev omogoča razpoznavo govora le iz posnetkov, ki so zapisani v formatu FLAC. Posledično je potrebno naš posnetek, pred pošiljanjem v obdelavo pretvoriti v zahtevani format. Posnetku se priložijo nekatere dodatne informacije, ki storitvi omogočijo boljšo razpoznavo. Med pomembnejšimi lahko izpostavimo jezik razpoznave ter format zapisa, v katerem naj bo vrnjen rezultat razpoznave (privzeto UTF-8). Vse dodatne informacije se v zapisu JSON priložijo k posnetku ter s pomočjo HTTP povezave pošljejo storitvi v postopek razpoznave.

Proces razpoznave - V tretji fazi razpoznave govora Google Speech Recognition storitev prejme naš paket in začne postopek razpoznave. Tu se poskuša s pomočjo kompleksnih algoritmov, ki skušajo posnemati delovanje nevronskih mrež, izločiti ter razbrati naš govor. Tu je uspešnost razpoznave odvisna tudi od prisotnih motenj ter jakosti govorjenja. V

času postopka razpoznave na strani odjemale aplikacije ne prihaja do čakanja, saj smo pri razvoju uporabili koncept niti. S pomočjo teh smo ločili proces razpoznave od glavne niti aplikacije, zaradi česar se lahko ta čas nemoteno izvajajo drugi procesi na izvajalnem seznamu.

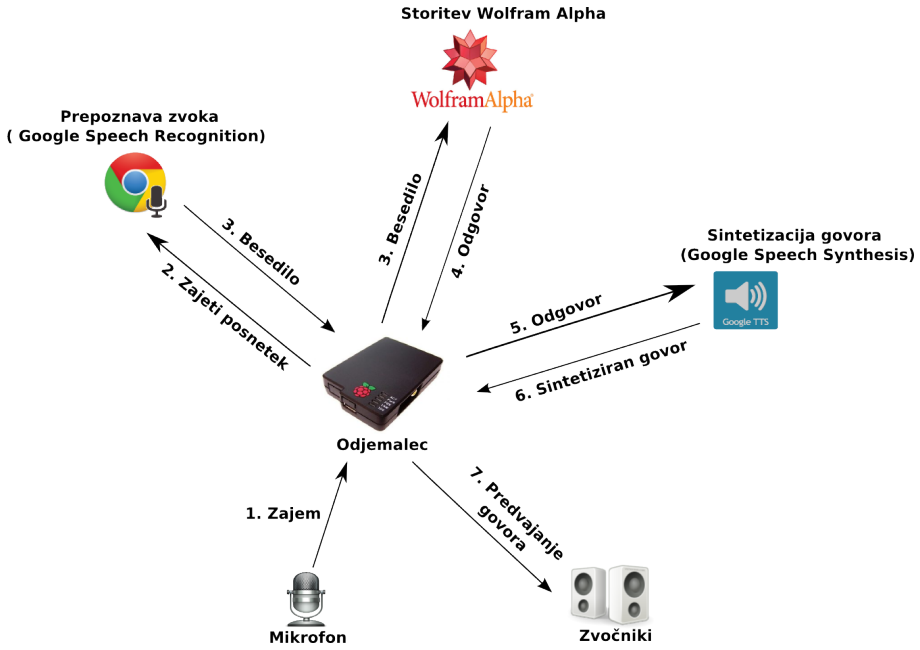
Rezultat - Ob koncu procesa razpoznave se nam vrne razpoznani govor v tekstovni obliki. Če je bila razpoznava neuspešna, se nam posreduje ustrezno opozorilo.

6.5 Glasovno upravljanje

Z implementacijo podpore razpoznavi ter sintezi govora, smo sistemu omogočili osnovne pogoje za upravljanje sistema s pomočjo glasu. Uporabniki lahko tako sistem upravljajo s pomočjo govora ter kot povratni odziv dobijo odgovor v sintetizirani obliki. S praktičnega vidika lahko tako vklapljamo/izklapljamo posamezne module oziroma izvedemo ponovni zagon ali izklop odjemalca. Kot je bilo že omenjeno, smo nabor možnosti dodatno razširili s podporo spletni storitvi Wolfram Alfa, ki nam zna poiskati odgovor na zastavljeno vprašanje v obliki govora. S praktičnega vidika uporabe bi lahko uporabnik zastavil vprašanje "Who are the Guns'n Roses music band"(Prevod: Kdo so glasbena skupina Guns'n Roses) nakar bi se kot odgovor preko zvočnikov predvajale informacije o omenjeni skupini.
Celotni postopek upravljanja smo predstavili v naslednjih postavkah:

Razpoznava govora - V prvi fazi se izvaja proces razpoznave govora, ki poskrbi za pretvorbo govora v besedilno obliko.

Krmiljenje - V drugi fazi se razpoznani govor posreduje zaledni logiki aplikacije, ki poskrbi za ustrezen odziv odjemalne aplikacije. Za razlikovanje ali gre za sistemski ukaz ali za vprašanje, se v sistemu uporabljajo določene rezervirane besede. Če ta zazna, da se je ukaz začel s katero izmed spodaj naštetih rezerviranih besed, potem se ta zaveda, da je treba izvesti sistemski ukaz.



Slika 6.3: Prikaz glasovnega povpraševanja

Activate 'ime_modula' - Aktivacija določenega modula.

Deactivate 'ime_modula' - Deaktivacija določenega modula.

Shutdown 'ime_odjemalca' - Izklop določenega odjemalca.

Reboot 'ime_odjemalca' - Ponovni zagon določenega odjemalca.

Če rezervirana beseda ni bila zaznana v razpoznanem govoru, se šteje, da je uporabnik odjemalcu zastavil vprašanje v obliki govora. Ta se nato pošlje storitvi Wolfram Alpha, ki bo poskušala poiskati odgovor nanj. Če je bilo povpraševanje uspešno, se aplikaciji nazaj posreduje odgovor v tekstovni obliki oziroma sporočilo s številko napake, v primeru neuspeha.
Glede na rezultat se nato pripravi povratni uporabniški odziv, ki se posreduje naslednji fazi upravljanja.

Povratni glasovni odziv V zadnji fazi se prejeti odziv pretvori v sintetizirano obliko človeškega govora, s pomočjo Google Speech storitve. Ta

se nato preko privzetega zvočnega izhoda predvaja na odjemalcu. V postopku predvajanja smo naleteli na težavo, saj je aplikacija zaznavala sintetiziran govor kot zvočne spremembe v prostoru ter jih poskušala razpoznati. Težava je nato vodila v neskončno zanko razpoznave ter predvajanja popačene razpoznave. Posledično smo problem rešili tako, da smo proces razpoznave ustavili v času predvajanja glasovnega odgovora.

Na sliki 6.5 smo predstavili upravljanje v primeru, da uporabnik zastavi neko glasovno vprašanje. Pri sistemskih ukazih je postopek podoben, vendar se korak, ki vključuje storitev Wolfram Alpha, preskoči ter namesto tega izvede sistemsko krmiljenje.

Za približno predstavo časovne kompleksnosti posameznih sklopov upravljanja smo izvedli dodatne meritve. Te smo prikazali v naslednjih dveh tabelah.

Tabela 6.1: Primerjava časovnih kompleksnosti glasovnih upravljanj

	Razpoznava govora	**Wolfram Alpha**	**Sinteza govora**	**Skupaj**
1	2.0	2.5	1.9	**6.4 s**
2	2.2	2.7	2.0	**6.9 s**
3	1.9	2.4	1.9	**6.2 s**
4	2.0	2.6	1.9	**6.5 s**
5	2.2	2.3	1.8	**6.7 s**
6	2.2	2.0	2.1	**6.4 s**
7	2.3	2.1	2.1	**6.5 s**
8	2.1	2.7	1.9	**6.7 s**
Povprečje	**2.11 s**	**2.4 s**	**1.95 s**	**6.54 s**

V zgornji tabeli 6.1 smo prikazali rezultate posameznih izvedenih glasovnih upravljanj. Iz rezultatov je tudi razvidno, koliko časa je potreboval posa-

mezni sklop v upravljanju za svojo izvedbo. Med drugim so tudi prikazani podatki o skupnem času izvedbe posameznega ukaza ter za vsak sklop posebej, koliko je v povprečju potreboval za svojo izvedbo.

V naslednji tabeli 6.2 smo prikazali podatke o številu poskusov, ki so bili potrebni za uspešno razpoznavo določenega glasovnega ukaza. V spodnjem delu tabele so tudi podatki o povprečnem številu potrebnih poskusov. Za primerjavo učinkovitosti smo uporabili naslednje glasovne ukaze:

- **1. Ukaz** - 'Activate red light' (Prevod: 'Vklopi rdečo diodo')
- **2. Ukaz** - 'Who are the beatles band' (Prevod: 'Kdo so glasbena skupina Beatles')
- **3. Ukaz** - 'Shutdown client raspberry' (Prevod: 'Izklop odjemalca raspberry)'

Tabela 6.2: Primerjava uspešnosti razpoznave različnih ukazov

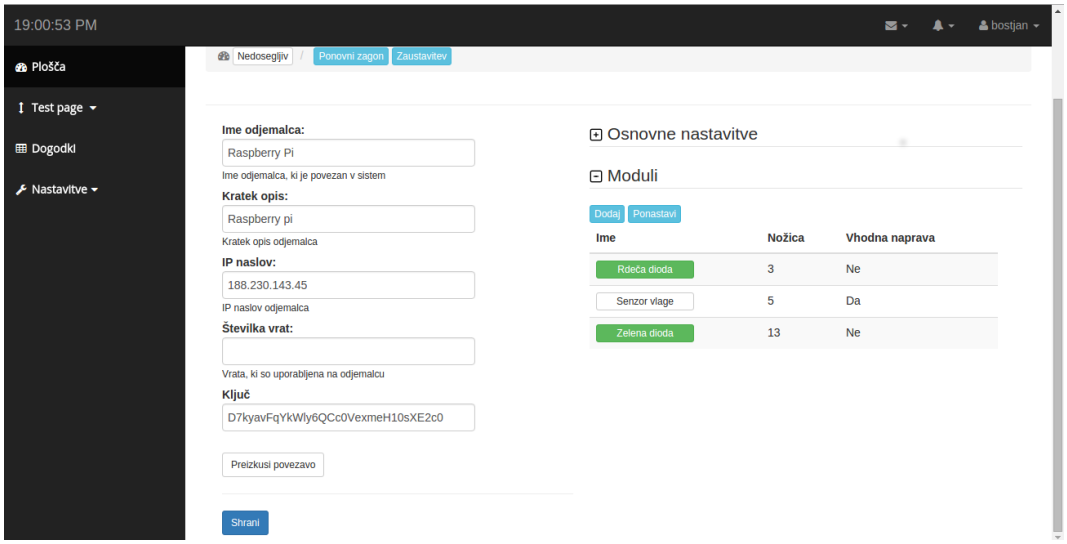
	Activate red light	**Who are the Beatles**	**Shutdown client**
1	2	2	2
2	1	2	1
3	2	1	1
4	1	2	2
5	1	3	2
6	2	1	1
7	1	2	1
8	1	3	1
povprečje poskusov	**1.37**	**2**	**1.25**

Iz števila potrebnih poskusov smo izračunali, da je bila uspešnost prvega in tretjega ukaza 72% med tem, ko je bila pri drugem 50%. Iz tega lahko

razberemo, da je v razmerju s kompleksnejšim ukazom sorazmerno slabša tudi njegova možnost razpoznave.

6.6 Prejete zahteve s strežnika

V ozadju aplikacije se neprekinjeno izvaja samostojni proces, ki preverja, ali so bile s strani strežnika poslane kakšne nove zahteve. Te so poslane v primeru, da se je na strani strežnika začel neki dogodek oziroma, če je uporabnik preko vmesnika ročno spremenil stanje modula, ki je otrok tega odjemalca. Prikaz ročnega vklopa preko vmesnika lahko vidimo na sliki 6.5. Ob vsaki poslani zahtevi se ji priložijo nekateri dodatni podatki (v zapisu JSON), s katerimi odjemalcu podamo potrebne informacije o spremembi stanja modula. V zapisu sta naslednja podatka:

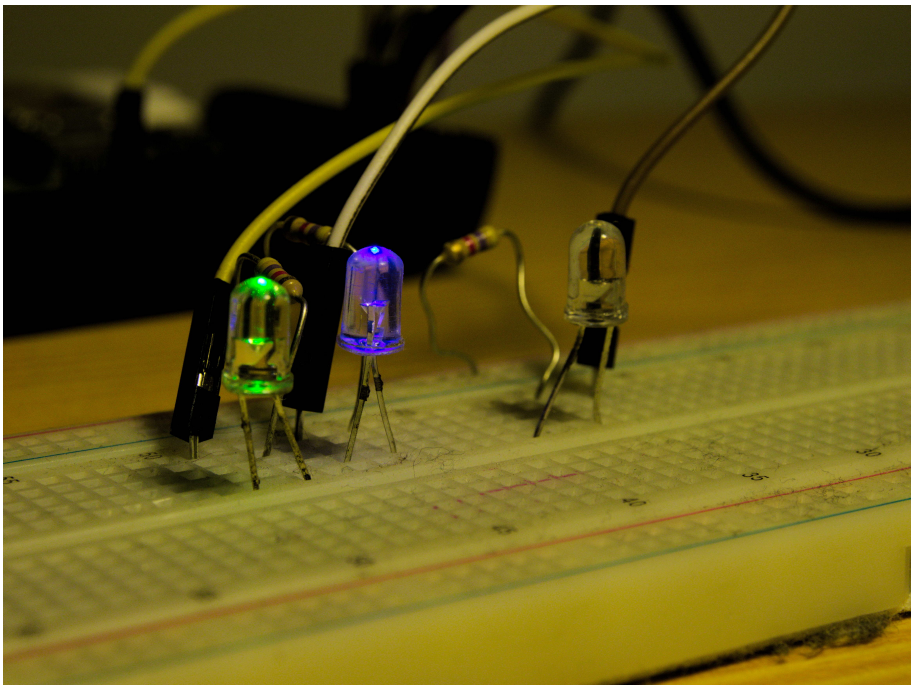


Slika 6.4: Prikaz aktiviranih modulov na strani strežnika

modul_ID - Označuje identifikator, ki je dodeljen fizičnemu modulu v podatkovni bazi na strani odjemalca. Ker so vsi identifikatorji unikatno določeni, se ta posledično preslikuje v GPIO nožico na vezju odjemalca.

Status - Hrani novo stanje, v katero naj preide fizični modul. Če je ta enak logični vrednosti 'True' bo modul prešel v delovanje oziroma v mirovanje, če bo ta enak 'False'. V primeru, da je nova vrednost enaka trenutnemu stanju modula, se nove spremembe ne upoštevajo.

Ob vsaki prejeti zahtevi aplikacija najprej preveri izvor pošiljatelja. S tem se dodatno prepričamo o verodostojnosti pošiljatelja ter preprečimo morebitne zlorabe. V postopku se primerja IP_naslov ter ključ pošiljatelja s tistim, ki je shranjen v lokalni podatkovni bazi. Priloženi podatki se nato pošljejo v obratni postopek deserializacije, katero smo podrobneje opisali v poglavju 2.7.2 (Zaščita poslanih podatkov), s čimer pridemo do izvornih podatkov pred opravljenim postopkom zaščite.



Slika 6.5: Prikaz aktiviranega modula

Aplikacija lahko nato s pomočjo priloženega atributa 'modul_ID' opravi povpraševanje v podatkovni bazi ter pridobi vse potrebne podatke o modulu, nad katerim želimo opraviti spremembo. Vsi podatki se začasno shranijo v

obliki objekta, s čimer je relativno preprosto izvesti spremembo trenutnega stanja modula.

Pri razvoju smo naleteli na težavo, saj smo za spreminjanje stanja nožice na Raspberry Pi računalniku uporabljali knjižnico RPi.GPIO [20], ki za upravljanje zahteva, da je uporabnik na strani odjemalca v sistemu prijavljen kot ROOT uporabnik. Ta uporabniški račun ima sistemsko privzet dostop do vseh datotek in pravic v operacijskih sistemih Linux. Z varnostnega vidika nam tak način upravljanja predstavlja varnostno grožnjo, saj bi s tem sistem izpostavili napadom s strani potencialnih napadalcev. Težavo smo rešili z uporabo knjižnice PiGPIO [21], ki nam med drugim omogoča uporabo nekaterih dodatnih funkcionalnosti, ki nas lahko obveščajo o vsaki uspešno opravljeni spremembi. Te smo uporabili tudi pri našem upravljanju nožic, saj se novo stanje modula v podatkovni posodobi šele ob potrditvi sprememb.

6.7 Varovala ob izpadu povezave

Spletna povezava je pomembni dejavnik, ki zagotavlja izmenjavo zahtev s strežnikom ter izvajanje glasovnega upravljanja. V primeru izpada bi prišlo do položaja, kjer bi bilo obnašanje modulov nenadzorovano ter v čakanju na ponovno dosegljivost povezave. Posledično bi lahko to vodilo do težav z dogodki, saj strežnik ne bi mogel sporočiti napravi, naj ob zaključku prestavi module v mirovanje. Kot rešitev smo vsaki odjemalni napravi dodali dodatne nastavitve, ki določajo njeno obnašanje ob odsotnosti povezave. Pomembnejše izmed vseh nastavitev smo podrobneje opisali v naslednjih postavkah:

Max_limit_odsotnosti - Največje dovoljeno število minut, po izteku katerih obravnavamo povezavo kot odsotno.

Odziv - Odziv modula na odsotnost povezave. Določimo se mu lahko, da se izklopi ali ostane v delovanju.

Kritični_protokol - Način na katerega nas bo odjemalec obvestil ob ponovni dosegljivosti povezave. Na voljo so prikaz obvestila na spletnem vmesniku ali elektronsko sporočilo.

Obvesti_uporabnike - Tu so v obliki seznama navedeni dodatni uporabniki, ki se jih naj obvesti ob ponovni dosegljivosti povezave. Znotraj seznama so uporabniki definirani z uporabniškimi imeni. V primeru, da ta v podatkovni bazi ne obstaja, se uporabnik preskoči.

Vrednosti parametrov je možno urejati preko spletnega vmesnika, v nastavitvah posameznega odjemalca. Te se prikažejo le, če so bile našemu uporabniškemu računu dodeljene pravice za njihov ogled ali urejanje. Za njihovo dodelitev je treba stopiti v stik s skrbnikom sistema.

Poglavje 7

Sklepne ugotovitve

V okviru diplomskega dela smo uspešno razvili sistem, ki omogoča povezovanje različnih tipov naprav v eno skupno mrežo. S končnim stanjem izdelka smo zadovoljni, saj smo uspešno izpolnili vse zastavljene cilje, ki smo si jih zastavili v postopku načrtovanja. Spletna aplikacija nam tako omogoča, da s priključenimi napravami upravljamo preko spletnega vmesnika, kjer lahko tudi spremljamo sveže informacije o delovanju sistema. Njen razvoj nam ni povzročal obsežnejših težav ter je potekal večinoma brez večjih težav. Na edino večjo oviro smo naleteli le ob razvoju namenskega dela aplikacije, ki skrbi za samodejne aktivacije naprav ob izbranih časovnih terminih. Težava se je pojavila v primeru, da smo za spreminjanje stanja naprave uporabili isto nit, kot jo uporablja glavna nit spletne aplikacije. Posledično je to vodilo do nepravilnih aktivacij naprav, kar smo rešili s prestavitvijo procesa v ločeno nit. Večji izziv nam je predstavljal razvoj odjemalne aplikacije, če smo natančnejši namenski del aplikacije, ki je zadolžen za glasovno upravljanje. Težave so se pojavile ob namestitvi določene verzije knjižnice PyAudio [26], ki jo aplikacija uporablja pri zajemu govora. Kot začasno rešitev smo uporabili starejšo verzijo knjižnice, ki bo v uporabi vse do izida novih popravkov.

V prihodnosti se bomo osredotočili predvsem na podporo možnosti namestitve odjemalne aplikacije tudi na druge tipe naprav. S teoretičnega vidika

bi nam odjemalca lahko predstavljal tudi telefon ali tablični računalnik s svojimi vgrajenimi senzorji. Tako bi nam vir podatkov lahko predstavljale meritve iz naslednjih senzorjev:

- merilec pospeškov,
- GPS lokacijski senzor,
- mikrofon,
- fotoaparat.

Zajete meritve bi lahko po naknadni obdelavi uporabili kot vir novih informacij, za spreminjanje stanj preostalih povezanih naprav v mreži. Tako bi lahko merilec pospeškov uporabili za aktiviranje določenih modulov, glede na izvedene premike telefona. S praktičnega vidika bi lahko tudi uporabili podatke iz GPS senzorja za vklop luči v stanovanju, ko bi ta zaznal, da se uporabnik nahaja v bližini doma. Med dodatne prožilce dejanj v sistemu bi lahko umestili tudi spletno storitev IFFT [22], ki omogoča povezovanje določenih storitev in naprav v verigo dogodkov. Ker je ta v spletnem oblaku, bi s tem posledično razbremenili naš sistem, saj bi vse preverjanje novih sprememb prestavili na stran spletne storitve. Verigi dogodkov lahko določimo situacije v katerih se naj proži, ter dejanje, ki naj ob sprožitvi izvede. Tako bi ji lahko določili, da nas z elektronskim sporočilom obvesti vsakič, ko bi nas naši prijatelji označili na Facebook sliki. Na strani sistema bi lahko izvedeno dejanje uporabili kot podlago za zvočno obvestilo na strani določenega odjemalca.

Glede na dejstvo, da je sistem trenutno v fazi preizkušanja ter pridobivanja povratnih uporabniških odzivov, lahko pričakujemo dodatne spremembe na strani vmesnika ter optimizacijo delovanja odjemalne aplikacije. Za končni produkt si želimo predvsem to, da bi bil preprost za uporabo ter hkrati eden izmed načinov poenostavitve vsakdanjih opravil.

Literatura

[1] L. Lamport. *Two Scoops of Django: Best Practices for Django 1.8*. Daniel Roy Greenfeld, Audrey Roy Greenfeld , 2015.

[2] Django documentation [Online]. Dosegljivo: https://docs.djangoproject.com/en/1.9/. [Dostopano 14. 11. 2015].

[3] Django REST framework [Online]. Dosegljivo: http://www.django-rest-framework.org/. [Dostopano 11. 12. 2015].

[4] Bootstrap documentation [Online]. Dosegljivo: http://getbootstrap.com/getting-started/. [Dostopano 11. 9. 2015].

[5] Celery official documentation [Online]. Dosegljivo: http://docs.celeryproject.org/en/latest/index.html. [Dostopano 12. 12. 2015].

[6] PostgreSQL database [Online]. Dosegljivo: http://www.postgresql.org/. [Dostopano 12. 9. 2015].

[7] Mysql database [Online]. Dosegljivo: https://www.mysql.com/. [Dostopano 12. 9. 2015].

[8] Heroku [Online]. Dosegljivo: https://www.heroku.com/. [Dostopano 12. 12. 2015].

[9] JSON format [Online]. Dosegljivo: http://www.json.org/. [Dostopano 12. 1. 2016].

[10] HTTP vs HTTPS protocol [Online]. Dosegljivo: https://www.instantssl.com/ssl-certificate-products/https.html. [Dostopano 11. 10. 2015].

[11] PBKDF2 cryptographic algorithm [Online]. Dosegljivo: https://en.wikipedia.org/wiki/PBKDF2. [Dostopano 28. 12. 2015].

[12] Raspberry Pi official documentation [Online]. Dosegljivo: https://www.raspberrypi.org/documentation/. [Dostopano 2. 8. 2015].

[13] Sublime Text editor [Online]. Dosegljivo: https://www.sublimetext.com/. [Dostopano 4. 12. 2015].

[14] Google SpeechRecognition package [Online]. Dosegljivo: https://pypi.python.org/pypi/SpeechRecognition/. [Dostopano 26. 12. 2015].

[15] Google Speech gTTS [Online]. Dosegljivo: http://blog.travispayton.com/wp-content/uploads/2014/03/Google-Speech-API.pdf. [Dostopano 6. 12. 2015].

[16] Speech synthesis [Online]. Dosegljivo: http://blog.teamtreehouse.com/getting-started-speech-synthesis-api. [Dostopano 4. 1. 2016].

[17] Raspberry Pi voice control[Online]. Dosegljivo: http://blog.oscarliang.net/raspberry-pi-voice-recognition-works-like-siri/. [Dostopano 4. 1. 2016].

[18] Wolfram—Alpha Webservice API Reference [Online]. Dosegljivo: http://products.wolframalpha.com/api/documentation.html. [Dostopano 4. 1. 2016].

[19] API— Wunderground weather [Online]. Dosegljivo: https://www.wunderground.com/weather/api/. [Dostopano 4. 1. 2016].

[20] RPi.GPIO library [Online]. Dosegljivo:
https://pypi.python.org/pypi/RPi.GPIO. [Dostopano 12. 12. 2015].

[21] PiGPIO [Online]. Dosegljivo:
http://abyz.co.uk/rpi/pigpio/. [Dostopano 12. 12. 2015].

[22] IFFT - Connect the apps you love [Online]. Dosegljivo:
https://ifttt.com/. [Dostopano 12. 12. 2015].

[23] SQLlite - Home page [Online]. Dosegljivo:
https://www.sqlite.org/. [Dostopano 3. 11. 2015].

[24] Extensible Markup Language (XML) [Online]. Dosegljivo:
https://www.w3.org/XML/. [Dostopano 12. 12. 2015].

[25] AJAX Programming (XML) [Online]. Dosegljivo:
http://api.jquery.com/jquery.ajax/. [Dostopano 26. 1. 2015].

[26] PyAudio library [Online]. Dosegljivo:
https://people.csail.mit.edu/hubert/pyaudio/. [Dostopano 26. 1. 2015].